uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías Industriales

Octubre 2017

Trabajo Fin de Grado

Desarrollo de aplicación Serious Game para rehabilitación motora

---

Autor

Carlos Tobarra Leal

Tutor

Edwin Daniel Oña Simbaña

Índice de Contenidos

Índice de figuras

Índice de tablas

Resumen

Este Trabajo Fin de Grado consiste en el desarrollo de un "juego serio" que aspira a ser una buena herramienta suplementaria o incluso un reemplazo para el fisioterapeuta en un proceso de rehabilitación. Este proyecto está centrado en el *Fugl-Meyer*, un test médico que evalúa la calidad del movimiento en pacientes que han sufrido un ictus. Por lo tanto, el objetivo principal es la evaluación de la evolución del paciente. Sin embargo, el juego también permite realizar actividades relacionadas con terapias de recuperación del movimiento, lo cual acelera las mejoras.

Para su realización, se propone el uso del sensor de *Microsoft 'Kinect'* y del software *Unity*, con los cuales se hace un análisis conjunto para valorar su precisión, exactitud y planos óptimos de medida.

Para concluir, destacar que la aplicación también será capaz de medir, recolectar y guardar información de las distintas sesiones. En consecuencia, al fisioterapeuta le será más fácil y le tomará menos tiempo juzgar el estado de los pacientes.

Palabras clave: Juego serio, Kinect, articulación, Unity, rehabilitación, ítem.

Abstract

This Final Project consists of the development of a serious game which aims to be a good supplementary tool or even a replacement for a physiotherapist in a rehabilitation process. This project is focused on the Fugl-Meyer, a clinical test which evaluates the quality of movement in patients who has suffered a stroke. Therefore, the main objective is the evaluation of the patient's progression. Nevertheless, the game also provides activities related with movement recovery therapies which expedite the improvements.

In order to make this game, we propose the use of the Microsoft Kinect sensor and the Unity software. With them, an analysis is made to evaluate the precision, the accuracy and the optimal plane of measure.

To sum up, this app also measures, collects and stores the data. Consequently, it should be easier and take less time for the physiotherapist to judge the status of the patients.

Key words: Serious game, Kinect, joint, Unity, rehabilitation, item.

1 Introducción

Este es el primer punto del Trabajo de Fin de Grado, por lo que se utilizará para realizar un acercamiento al mismo. Se expondrá la motivación que ha llevado a realizarlo, los conocimientos previos que se tenían sobre la materia y los objetivos que se buscan con la elaboración de este proyecto. Por último, se comentará la estructura en la que se dividirá el trabajo, para facilitar la comprensión a los lectores.

1.1 Motivación

Durante las últimas décadas, la tecnología ha aumentado exponencialmente y todas las predicciones pronostican que el ritmo no va a cambiar, al menos por el momento. Luego, que nos impide soñar con la existencia de todos esos instrumentos de las películas de ciencia ficción. Ciertamente, nada. Sin embargo, no se van a llevar a cabo si todo el mundo sueña con ello. Para que la tecnología llegase hasta el punto en el que está hoy en día, muchas personas han tenido que trabajar en proyectos poco realistas, pero necesarios. Ya que sentarían las bases en la actualidad. Por lo tanto, es una tendencia que interesa conservar.

La finalidad de este proyecto es dar un pequeño paso en la ausencia del especialista durante el proceso de rehabilitación, una realidad aún lejos de conseguirse. No obstante, los avances de la tecnología ya permiten que este se vea reemplazado en ciertos aspectos de la misma. Gracias a los sensores de movimiento se puede rastrear al paciente, y gracias a los motores gráficos, se puede crear un videojuego sin ser un experto en el tema. Esto puede permitir que en las partes donde el paciente se mueve solo, el terapeuta pueda ausentarse, ya que el *Serious Game* se encargaría de ello. En el futuro, con la ayuda de robots, se podrían llegar a realizar el resto de tareas, como puede ser el ofrecer un objeto al paciente con cierta resistencia o el mover las articulaciones del paciente para comprobar su movilidad.

Por tanto, la principal motivación que me ha llevado a realizar el proyecto es la de aportar mi grano de arena. Contribuir en el desarrollo de nuevas tecnologías que hagan nuestra vida más sencilla.

Sin embargo, hay que destacar otras motivaciones que también me han ayudado a decidirme por este proyecto. Siempre me he sentido atraído por la programación y por los videojuegos, por lo que realizar una labor en la que los dos estén implicados es algo que me atraía. La curiosidad por el funcionamiento del sensor también ha tenido parte de culpa. Pero lo realmente destacable es la idea de poder ayudar a personas con problemas médicos con los conocimientos que he obtenido durante el grado. Cuando elegí la carrera universitaria, dudé entre la rama de biología y la de tecnología, por lo que volver de cierta forma a las raíces lo hace doblemente gratificante.

1.2 Conocimientos previos

Para realizar un proyecto de estas características, en la que la mayoría del contenido se realizará mediante código, son necesarios unos conocimientos elevados de programación. En mi caso, antes de comenzar, ya había trabajado con varios lenguajes de programación, entre los cuales cabe destacar C++, puesto que es el lenguaje del que disponía más experiencia. Sin embargo, el proyecto requería conocimientos de C#.

Esto no provocó un gran problema, ya que ambos lenguajes son similares, lo que permitió que el proceso de aprendizaje de C# no fuera tan largo y complejo como lo suele ser. Para adquirir los conocimientos y la fluidez necesaria en este lenguaje, se siguieron varios tutoriales online.

Varios de ellos, fueron los que se ofrecen en la página web oficial de Unity. Estos, además, sirvieron para aprender los fundamentos del motor gráfico con el que se crearía el videojuego. En este apartado, no se disponía de ningún conocimiento. La única tarea efectuada que tenía alguna relación con el mundo de los videojuegos fue la creación de varios mini juegos simples, como el *"pong"* o *"hundir la flota*", cuyo objetivo no era más que el de la práctica de la programación.

Por tanto, se tuvieron que dedicar varios meses a descubrir las características que ofrecía el motor. Como ya se ha dicho, se siguieron varios tutoriales de la página oficial, además, se realizaron varios juegos cuyas mecánicas podrían aplicarse al *Serious Game* final. No obstante, debido a la gran cantidad de opciones que ofrece el *software*, ha sido necesario acudir a la documentación con frecuencia.

En cuanto al dispositivo de captura de movimiento, el sensor *Kinect* de *Microsoft*, no se disponían conocimientos de él, más allá de los que un usuario promedio tiene. Es decir, se sabía de su existencia y de su comercialización con la videoconsola *Xbox*, pero se desconocía todo lo relacionado con su parte técnica. Por ello, hubo que investigar en varios foros como empezar a usar este aparato. El software ofrecido por *Microsoft*, *Kinect for Windows SDK*, también ayudó en el aprendizaje, ya que ofrece varios ejemplos sobre cómo utilizar el sistema.

1.3 Objetivos

El presente Trabajo Fin de Grado consiste en el desarrollo de un *Serious Game* que ayude en el proceso de rehabilitación de un paciente que haya sufrido un ictus. Para ello, el juego ha de poseer varias funciones.

La función principal es la de poder realizar parte del test *Fugl-Meyer*, concretamente la relacionada con la función motora del miembro superior. El juego pedirá al paciente que realice los movimientos de esta parte del test, y con la ayuda del sensor de movimiento *Kinect*, se rastrearán las posiciones de las articulaciones del paciente. Esto permitirá obtener unos resultados precisos sobre el progreso y las limitaciones del paciente.

Sin embargo, esta no será la única función del juego. Este también proporcionará un entorno en el que el paciente pueda mejorar sus habilidades, es decir, se le ofrecerán mini juegos diseñados especialmente para practicar los movimientos que se están evaluando. Hay que destacar que, al realizarse todo el proceso mediante un videojuego, se pretende añadir un toque de diversión y competitividad. Con esto se intenta que el paciente se sienta más motivado y sea más constante, ya que son dos cualidades clave para una próspera rehabilitación.

Por último, remarcar la cantidad de tiempo que se podrían ahorrar los terapeutas, ya que no es necesario que estén presentes mientras el paciente juega al *Serious Game*. Esto es una gran ventaja frente a otro tipo de métodos de rehabilitación. Si a esto se le suma el hecho de que el sensor que se va a usar es de bajo coste, obtenemos un resultado final con unas características económicas que favorecen su uso real en el futuro. Es importante destacar, que habría que efectuar muchas más pruebas de las realizadas, comparando las medidas con otros sistemas, para considerar los resultados obtenidos viables.

1.4 Marco regulador

Tras clarificar el objetivo que se busca, se puede deducir que este proyecto no estará sujeto a una gran cantidad de reglamentos. A primera vista, los únicos que se deberán tener en cuenta son aquellos relacionados con la interacción del paciente, la obtención y guardado de datos y el uso adecuado de las tecnologías utilizadas. Cabe destacar que el presente Trabajo Fin de Grado se realiza de acuerdo con el Real Decreto 1393/2007.

Legislación

Después de realizar una búsqueda sobre la legislación aplicable al proyecto, se han encontrado varias cualidades que este debe cumplir. En primer lugar, se debe evitar cualquier tipo de riesgo en el paciente. Para ello, se han de desarrollar unos movimientos en los que la integridad del paciente no se vea afectada. Esto podría pasar, por ejemplo, si un movimiento pone a prueba el equilibrio del paciente, este podría desequilibrarse y provocarse lesiones. Por tanto, o no se desarrollan movimientos con riesgo potencial o se pide al paciente que primero firme un documento en el que se comprometa a no realizar esos ejercicios sin vigilancia o ayuda.

En el caso de que se incumplan las condiciones mencionadas y el usuario del *Serious Game* sufra alguna lesión, según el artículo 1902 C.C “El que por acción u omisión causa daño a otro interviniendo culpa o negligencia, está obligado a reparar el daño causado”. Por tanto, sería necesario la adquisición de un seguro de responsabilidad civil, para hacer frente a la posible indemnización por los daños causados.

Dado que se van a pedir datos personales al paciente y se va a adquirir y guardar información sobre las características de cada sesión, es imprescindible acatar la ley orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.

Según esta, la información que provea el paciente no podrá ser transferida sin su consentimiento y solo será usada con fines médicos.

Estándares técnicos

En cuanto a los estándares técnicos, apenas se tienen en cuenta por las propiedades del proyecto. Habría que destacar los *drivers* proporcionados por *Microsoft* para el desarrollo y funcionamiento de aplicaciones que usen *Kinect* en ordenador. Aparte de esto, realmente no hay ningún tipo de estandarización en el desarrollo de *Serious Games*. Se podría hablar de motores gráficos más populares, de sensores más usados para este tipo de proyectos o de lenguajes de programación, pero dado que la elección de estos depende en gran medida de las características que se busquen y los gustos personales, no se puede considerar como estándares técnicos.

Propiedad intelectual

Esta consiste en el conjunto de derechos que corresponden al autor sobre su obra. En un videojuego normal, se suelen registrar de forma individual las distintas partes, es decir, el guión, el software, los personajes… En nuestro caso, esto no se hará, ya que al tratarse de un Trabajo Fin de Grado sigue un procedimiento diferente. Los TFGs son trabajos protegidos por la Ley de Propiedad Intelectual del Real Decreto Legislativo 1/1996, en 12 de abril. Según esto, la propiedad intelectual le pertenece al alumno que lo haya realizado.

No obstante, es importante destacar que la Universidad Carlos III de Madrid ofrece tres opciones al estudiante. En primer lugar, puede publicar su trabajo obteniendo las condiciones de uso de la licencia *Creative Commons*. En cuyo caso será de acceso libre y gratuito con las condiciones de que se reconocerá la autoría, se usarán con fines no comerciales y se compartirá de igual forma, es decir, con la misma licencia. En segundo lugar, se puede publicar de forma restringida, permitiendo el acceso solo a la comunidad universitaria. Por último, se puede denegar la publicación del proyecto, siempre y cuando se justifique mediante una posible patente o un trabajo financiado por una empresa. En el caso que nos atañe, se elegirá la primera opción, por lo que todo el proyecto será público bajo una licencia de *Creative Commons*.

1.5 Estructura

La memoria se dividirá en ocho puntos, de los cuales cinco tratarán el tema del trabajo, uno el entorno y dos servirán como introducción y conclusión del problema. En el siguiente apartado se iniciará el tema central del proyecto introduciendo al lector al mundo de los videojuegos. Se darán las pautas necesarias para que se clarifique el significado de S*erious Game*, y los objetivos que se buscan con él.

A continuación, se hablará de todo el tema médico. Este capítulo estará compuesto de dos secciones. En la primera se darán todos los detalles posibles sobre la patología que se va a tratar, es decir, sobre el ictus, incluyendo métodos de rehabilitación utilizados

actualmente. Mientras que en la segunda se van a exponer las características de la escala *Fugl-Meyer*, que será la que se implementará en el juego serio.

En el siguiente punto, se expondrá todo lo relacionado con las tecnologías a usar para la elaboración del juego. En primer lugar, se analizarán las distintas formas de desarrollar videojuegos en la actualidad. Posteriormente, se evaluarán los sensores de movimientos más populares en el mercado. Finalmente, se hará una comparación entre las diferentes opciones con tal de elegir la mejor combinación posible de tecnologías.

En el capítulo 5 se explicará todo lo concerniente al desarrollo del proyecto. Por tanto, se comentarán todos los pasos relevantes que se han dado, tanto con el motor gráfico, como con el sensor. Esto incluye todo lo relacionado con la programación y el análisis de datos. A continuación, en el apartado 6, se mostrarán las pruebas realizadas y, teniéndolas en cuenta, se explicarán que partes del test se han considerado válidas y cuales no han conseguido incorporarse al resultado final.

En el siguiente apartado se tratará el entorno socio-económico, dividiéndose este en el impacto que puede producir el resultado del proyecto en diferentes ámbitos y en la previsión de costes. Para finalizar, en el último punto se realizará una conclusión en la que se analizarán los resultados obtenidos. También se detallarán los problemas enfrentados y las posibles mejoras que se podrían implementar en un futuro.

2 Videojuegos

Como ya se ha explicado, el proyecto consiste en el desarrollo de un *Serious Game*. Por tanto, es necesario explicar ciertos aspectos sobre él. En este apartado se tratará en profundidad en que consisten, como surgieron, cuáles son sus objetivos y los tipos más importantes que existen.

Además, ya que los *Serious Game* surgen como resultado de la evolución de los videojuegos, es preciso, en primer lugar, analizar estos. Para ello, se tratará de dar la información suficiente al lector, como para que este tenga los conocimientos necesarios sobre la industria y como esta ha desembocado en la creación de los "juegos serios". Con este objetivo, se expondrá la historia de los videojuegos y los diferentes tipos existentes en la actualidad.

Pero, antes de continuar, es necesario aclarar que es considerado un videojuego.

Los videojuegos son aplicaciones interactivas orientadas principalmente al entretenimiento, en las cuales una o más personas interactúan, mediante algún controlador, con un dispositivo capaz de mostrar videos. Habitualmente, el dispositivo electrónico que permite jugar es conocido como plataforma y va desde ordenadores, pasando por videoconsolas, hasta teléfonos móviles. En cuanto al controlador, este puede ser un mando, un teclado, una pantalla táctil o, en general, cualquier dispositivo que permita interactuar con el videojuego. En nuestro caso será un sensor de movimiento.

Una vez que sabemos esto, es hora de comenzar a examinar la industria, empezando por su historia.

2.1 Historia de los videojuegos

A lo largo de los años se ha intentado concordar sobre cuál fue el primer videojuego, pero debido a las diferentes interpretaciones sobre que se puede considerar uno y que no, nunca se ha llegado a un acuerdo.

Sin embargo, si que se puede coincidir en que los inicios de los videojuegos se retornan a finales de la década de los 40. En esta época surgió lo que podría considerarse como la base de los primeros videojuegos. Por ejemplo, en 1947, *Thomas T. Goldsmith* y *Estle Ray Mann* crearon un sistema electrónico que simulaba el lanzamiento de misiles, funcionaba mediante válvulas y no había movimiento en pantalla, por lo que no se considera un videojuego. Otro ejemplo podría ser *Alan Turing*, que en colaboración con Champernowne, escribió un programa para jugar al ajedrez.

No fue hasta 1952, cuando apareció el 'Cruz y Raya', desarrollado por *Alexander s. Douglas*. Este era una versión digital del popular juego con su mismo nombre. Cierta parte de la comunidad lo considera el primer videojuego, pero al no tener movimiento en pantalla (solo aparecen o desaparecen imágenes), la mayoría no lo consideran como tal.

Posteriormente, en 1958, *Tennis for Two* fue desarrollado por el físico norteamericano. Este consistía en una línea horizontal y otra vertical que representaban la pista de tenis y la red respectivamente. Los jugadores podían elegir el ángulo con el que se dispararía la pelota y el momento en el que golpearla. Este juego se llevó a cabo con un programa de cálculo de trayectorias y un osciloscopio.

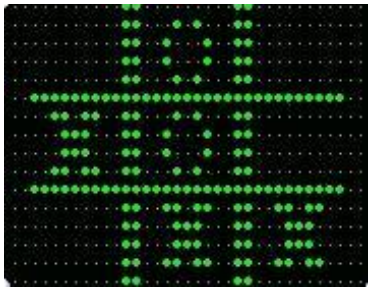

Figura 2.1: Tres en raya

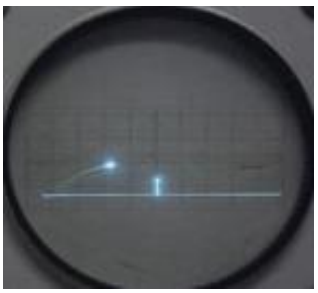

Figura 2.2: Tennis for Two

Tiempo después, en el *Massachussets Institute of Technology*, varios estudiantes liderados por *Steve Russell*, crearon *Spacewar!*. El juego consistía en dos naves, cada una manejada por un jugador, que luchaban entre ellas mediante disparos. Funcionaba en un PDP-1, un superordenador de la época. Puede considerarse como el primer juego para ordenador, ya que el anterior funcionaba mediante circuitos, no en un ordenador. Tuvo bastante éxito en el ámbito universitario, pero no salió de ahí, ya que el equipo necesario para jugar tenía un coste de 120000$.

Este problema sería solventado tiempo después por *Nolan Bushnell*, quien con el tiempo fundaría *Atari*. Nolan creó su propia versión de *Spacewar!* y la llamó *Computer Space*. Con la sustancial diferencia de que no hacía falta un PDP-1 para jugar. Para ello diseñó un dispositivo especializado, que no tenía tantas funciones como un ordenador y que, por tanto, redujo considerablemente los costes. A pesar de todo, no tuvo el éxito suficiente.

Figura 2.3: Spacewar!

Figura 2.4: Computer space

A partir de aquí, comenzó el auge de los videojuegos. En 1972, fue lanzada la primera consola de la historia, la *Magnavox Odyssey*. Su creador, el ingeniero Ralph Baer, es considerado el inventor de los videojuegos tal y como los conocemos. La consola tenía

la capacidad de crear una línea central, dos puntos y una pelota. Esto, junto con una serie de elementos que se vendían junto a la consola como láminas transparentes o cartas, permitía jugar hasta 12 juegos diferentes.

Nolan se inspiró en esto y en el ping-pong para crear *Pong*, considerado por muchos el primer videojuego de la historia. Este juego consistía en dos rectángulos, cada uno controlado por un jugador, y una pelota, la cual no podían dejar que pasase. Se jugaba en una máquina recreativa, se distribuyó en bares, salones recreativos y aeropuertos, y debido a su gran éxito, propició un desarrollo desorbitado de la industria [1].

Figura 2.5: Pong

Durante los siguientes años hubo unos avances tecnológicos muy importantes en el terreno de los microprocesadores y los chips de memoria. Además, aparecieron otros juegos que gozaron de gran éxito en los salones recreativos, como *Space Invaders* (Taito) o *Asteroids* (Atari). Esto consolidó la industria y favoreció a la popularización de las consolas domésticas como la Atari 2600, las cuales se harían con la mayoría del mercado en el futuro.

A principios de los años 80, el mercado doméstico estaba cogiendo fuerza con consolas como *Oddysey2, Intellivision, Colecovision o Atari 5200* y ordenadores como *ZX Spectrum o Commodore 64*. Sin embargo, seguían dominando las recreativas, en las que triunfaron juegos como *Pacman (Namco) o Tron (Midway)*. También fue durante estos años cuando Nintendo empezó a comercializar consolas y creó a Mario, quien sería la marca de la compañía hasta la actualidad, para el juego *Donkey Kong*.

A pesar del crecimiento del sector, hubo un grave problema que desembocó en una crisis para la industria del videojuego, la cual estuvo a punto de desaparecer en Norteamérica. El problema se podría dividir en varios puntos. En primer lugar, cualquier persona podía crear un videojuego, ya que las empresas no cobraban tarifas por desarrollar en sus consolas. Esto propició una bajada de la calidad media en los juegos. Todas las empresas quisieron participar en el boom de la industria, por lo que se creó una saturación de consolas en el mercado. A todo esto, hay que añadirle lo más importante, no había información suficiente para el consumidor, por ser un sector nuevo. Esto provocó la pérdida de confianza del consumidor, el cual no sabía que productos merecían la pena y cuáles no. Todo esto desembocó en una bajada de precios de los comerciantes para poder dar salida al stock, lo que irremediablemente propició la crisis.

Esta crisis no fue sufrida en Japón, puesto que en este territorio triunfó la *fanicom* por encima de la competencia, evadiendo así la saturación del mercado. De hecho, esta consola fue la salvadora de la industria en 1985, al estrenar *Super Mario Bros*, que vendió más de 10 millones de copias y reavivó el sector. En occidente era conocida como *NES (Nintendo Entertainment System*) [2].

A partir de aquí, surgieron otros sistemas como la *Master System* de *Sega* o la *Atari 7800*, que gozaron de popularidad y dieron fruto a clásicos como el *Tetris*. A principios de los 90 las consolas dieron otro salto evolutivo con la generación de los 16

bits, en la cual destacan la *Super NES y la Mega Drive*. Cabe remarcar que esta generación casi igualaba en potencia a las máquinas recreativas y favoreció a la introducción del CD-ROM.

En la siguiente generación, la llamada de 32 bits, se dio otro gran avance en cuanto a las posibilidades ofrecidas por las consolas, permitiendo los juegos en 3D. En esta generación destacan *Sega saturn, Nintendo 64, Atari Jaguar y Play Station*. A partir de este momento, las recreativas comenzaron un declive lento, pero imparable. La tecnología ya había avanzado lo suficiente como para ofrecer calidad a un precio asequible. Esto se hace aún más evidente con la aparición de las consolas portátiles alrededor del año 2000. El mercado se divide en un principio entre varias empresas, pero finalmente se decanta por la *Game Boy* de *Nintendo*, la cual acapara el mercado portátil en su totalidad.

Figura 2.6: Consolas de 5ª generación

En cuanto a los juegos más destacados de estas generaciones destacan las franquicias de *Final Fantasy, Resident Evil, Gran Turismo, Metal Gear Solid, Pokémon* y *Super Mario*.

El comienzo del año 2000 se considera el inicio de la era moderna. En esta, las opciones ofrecidas por las empresas disminuyeron, favoreciéndose así la especialización de estas compañías. Como resultado solo destacaron tres consolas: la *PlayStation 2*, la *GameCube* y la primera *Xbox*. Además, también comenzó a crecer el uso del ordenador, lo que favoreció la aparición de más juegos en la plataforma

Cabe remarcar la aparición de *Steam* en 2003, que fue fundada por la empresa estadounidense *Valve Corporation*. Es una plataforma de distribución digital de videojuegos únicamente para pc. Además, también se encarga de gestionar los derechos y los servicios multijugador. El primer juego es salir en la plataforma fue *Counter Strike 1.6*, un superventas de la compañía. Actualmente, es la empresa líder en el mercado digital y maneja cantidades inmensas, tanto de usuarios registrados como de catálogo disponible.

Si volvemos al halo de las consolas, se podría destacar en 2005 la aparición de la Nintendo DS, la consola portátil más vendida de la historia. En cuanto al formato de sobremesa, se anuncia una nueva generación. Estará formada por la *Xbox 360, la PlayStation 3* y la *Wii*.

Figura 2.7: Consolas de 7ª generación

Desde este momento y hasta la actualidad, las empresas dominantes del mercado siguen siendo las mismas, solo que con versiones mejoradas de sus dispositivos anteriores. Estos vendrían a ser la *PlayStation 4, la Xbox One, la Nintendo Swich o la Nintendo 3DS XL*. Hay que destacar que el mercado de las consolas está perdiendo una gran cantidad de usuarios, que se está desplazando hacia los móviles y los ordenadores. Dispositivos del día a día que tienen la potencia necesaria como para ejecutar juegos con altas cargas gráficas [3].

Hay que remarcar la entrada de la realidad virtual (VR) en el mercado de los videojuegos. Empresas como Sony ya la ha implementado en su consola, y ciertos móviles permiten su uso gracias a un dispositivo especializado. Actualmente sus ventas siguen siendo muy bajas, pero conforme evoluciona la tecnología, es esperable que, en el futuro, puedan ser fundamentales para la industria.

Se podría concluir con que el mercado está cambiando de tendencia, lo que podría provocar la desaparición de las consolas, en favor de otras plataformas más flexibles, como son los móviles.

2.2 Clasificación de los videojuegos

Hoy en día, la cantidad de juegos existentes es tan inmensa, que es prácticamente imposible clasificarlos todos. Además, no hay un solo tipo de organización, dependiendo de la fuente, esta puede variar. Los factores diferenciadores más comunes para tener en cuenta son la edad recomendada del juego, la representación gráfica, el tipo de interacción con el jugador, las mecánicas de juego, su tipo, la plataforma y los controles que se requieran.

En nuestro caso, la clasificación se va a realizar teniendo en cuenta el tipo de juego que es, es decir, según su género. Esto implica agrupar todos los que posean una serie de elementos comunes en el sistema de juego. Aun así, hay que remarcar que muchos videojuegos poseen elementos de varios géneros, por lo que no se pueden clasificar en un solo grupo. En adición a esto, hay que recalcar que, al no haber una clasificación oficial concordada entre la mayoría, el catálogo que se va a exponer a continuación puede variar en función de la fuente consultada [3] [4].

Acción

Este tipo de juegos se caracterizan por poner a prueba la destreza, agilidad, velocidad y tiempo de reacción del jugador. Dentro de este género existen muchos subgéneros:

- Plataformas. Probablemente uno de los más conocidos debido a la fama de Mario. Consisten en el manejo de un avatar que debe avanzar por un escenario lleno de obstáculos y/o enemigos. Este género destaca por la fluidez y el movimiento del personaje. Se pueden crear juegos de este tipo tanto en 2D como en 3D.
- Lucha. Este género se hizo famoso en la época de las máquinas recreativas. Radica de dos avatares que luchan entre ellos, normalmente en un entorno en 2 dimensiones, aunque se puede dar el caso de que haya 3D y varios personajes a la vez. Suelen usar artes marciales, como en *Street fighter*, o poderes especiales, como en *Dragon Ball*.
- *First Person Shooter(FPS).* Como su propio nombre indica, tratan de disparar en primera persona. Esto significa que la cámara está situada de tal forma, que el usuario vea lo que vería el avatar, dando así la impresión de que es el propio jugador el que está inmerso en el mundo virtual. Su objetivo suele ser el de eliminar a otros personajes, predominando jugabilidad y premiando los reflejos del usuario. El más importante a día de hoy es el *Call of Duty*.
- *Third Person Shooter(TPS).* Similar al anterior, pero con la diferencia de que en esta ocasión el avatar es visto desde atrás. Esta diferencia hace que sean juegos en los que se sacrifica la rapidez por la interacción por el entorno. Se puede ver como el personaje realiza las acciones, como coberturas, saltos… Podría destacarse la saga de *Uncharted*.

- *Beat 'em up.* También llamados de lucha a progresión. Los usuarios deben luchar con hordas de enemigos que van aumentando su poder según se va avanzado de nivel. Este género gozó de gran éxito en las máquinas recreativas. Destaca *The Warriors*. Una variación de este género es el *Shoot 'em up*.
- Sigilo. Este género se basa en la discreción del usuario. Es fundamental la estrategia, ya que el objetivo principal suele ser el de superar diferentes escenarios repletos de enemigos sin que estos se percaten de tu presencia. El juego estrella de este género es el *Metal Gear*.

Arcade

Se caracterizan por una acción simple y rápida. Destaca la jugabilidad por encima de cualquier otro elemento del juego. Este género tuvo su época dorada en los años 80. No requiere de historia, solo unas mecánicas que inciten a la repetición. Grandes representantes de este género son el *Pac-Man, Space Invaders o Asteroids*.

Deporte

Este tipo se basa en la representación digital de un deporte. Existen juegos de futbol, baloncesto, tenis, golf, hockey y un largo etc. Incluso hay juego como el *Wii Sport,* que son una recopilación de varios. En estos ha de destacar la jugabilidad, debiendo ser divertida y cómoda. El juego que más éxito ha recabado hasta la fecha es el *FIFA*.

Carreras

Se basan en recorrer una distancia más rápido que los adversarios. Normalmente este tipo de juegos es de coches, es decir, las carreras se hacen con vehículos. El jugador es el encargado de manejar uno, en primera o tercera persona, y puede competir contra si mismo en contrarreloj o contra más jugadores. Los juegos más conocidos de este género son el *Mario Kart y Gran Turismo*.

Simulación

Es un género muy variado que consiste en imitar aspectos de la vida real. En la mayoría de juegos el jugador se pone en la piel de su avatar y las situaciones que se dan son totalmente posibles en la realidad. Dentro de esta categoría se puede distinguir varios subgéneros:

- Simulación de guerra. Este tipo de videojuegos se crearon con el objetivo de entrenar al ejército. Sin embargo, con el tiempo, se ha ido adentrando en el público general. Algunos juegos destacados son *Operation flashpoint* o *Arma*.
- Simulación de vida. Se centran en recrear la vida de las personas. Los juegos de este tipo suelen consistir en tomar decisiones que afecten de un modo u otro a tu personaje. Este tiene necesidades y emociones, de esta forma interactúa con el jugador, el cual tiene que controlar todos los aspectos de su vida. El más famoso de este género es *Los Sims*.
- Simulación de vehículos. Este subgénero se ha popularizado en los últimos años. Puede consistir desde la simulación realista de carreras, hasta la administración de un taller en el que hay que reparar coches todos los días. Destaca el *Project Cars.*
- Simulación de construcción. Como su propio nombre indica consiste en construir desde piezas hasta proyectos. Los juegos más importantes de este género son los que tienen como objetivo la creación y el mantenimiento de una ciudad. En estos se ha de tener en cuenta multitud de factores, como el terreno, el costo de los materiales, el tiempo… Destacan como juegos más importantes *Age of Empires y Cities: Skyline*.

Educativo

Son aquellos que aprovechan la idea de diversión asociada a los videojuegos para enseñar. Su principal objetivo para de ser el de divertir y entretener a ser el de educar, pero sin dejar estos totalmente de lado. La mayoría de estos juegos están orientados a los niños pequeños.

Rol

Más conocidos como *Role-Playing Game (RPG),* se basan en una historia profunda, en la cual el personaje o personajes principales, manejados por el jugador, van evolucionando. Esta evolución se da tanto en la personalidad como en las habilidades de lucha. También destacan la cantidad de personajes secundarios y la interrelación existente entre todo el mundo. Para contener todo esto se crean mapas inmensos, en los cuales se premia la exploración. Normalmente, este género, suele poseer armas, magias, artes… en general, todo tipo de habilidades con las que luchar.
En los RPG clásicos el combate solía establecerse por turnos. Cada cierto tiempo, un personaje puede realizar una acción, como luchar, usar una magia o curarse. Mientras se decide que hacer el combate está en pausa, una vez decidido, se realiza la acción y el tiempo sigue hasta que le toque a otro personaje. Hoy en día el combate suele ser en tiempo real, es decir, se ataca y se defiende a la vez.
Cabe destacar que se han creado una gran cantidad de subgéneros, entre los cuales destacan los *MMORPG (Massive Multiplayer Online RPG*). Estos se caracterizan

por jugarse online. Cada jugador se crea un avatar personalizado, con el que puede interactuar con otros jugadores. El juego más destacado es *World of Warcraft*.

Aventura

Fueron, junto a los arcades, los primeros en lanzarse al mercado. Tuvieron su auge entre los 80 y los 90, aunque desde hace unos años están volviendo a obtener algo de interés. Se caracterizan por contar una historia, en la que el jugador encarna al protagonista y toma las decisiones por él. Por lo general, debe resolver una serie de incógnitas y rompecabezas para avanzar en la historia. Primero surgieron las aventuras textuales y, posteriormente, las aventuras gráficas. Las primeras se caracterizan por escribir la acción a realizar o señalarla con el teclado entre una lista de acciones. Las segundas son del tipo *Point and click*, en las que el jugador selecciona objetos en la pantalla para interactuar con ellos. Algunos juegos a destacar de este género son *Heavy rain o The secret of monkey island*.

Para finalizar, se va a intentar examinar la popularidad de cada género. Para ello, se va a tener en cuenta que cada uno tiene una cuota de mercado que varía en función de la calidad de los juegos sacados y del interés del consumidor en ellos. En 2014, *Entertainment Software Association* realizó un estudio en Estados Unidos, segundo país con más ingresos por videojuegos, en el que se analizó el porcentaje de mercado que acaparaba cada género. Los resultados mostrados en la **figura 2.8** muestran cómo, hoy en día, los videojuegos más populares son los de acción y los de deportes.

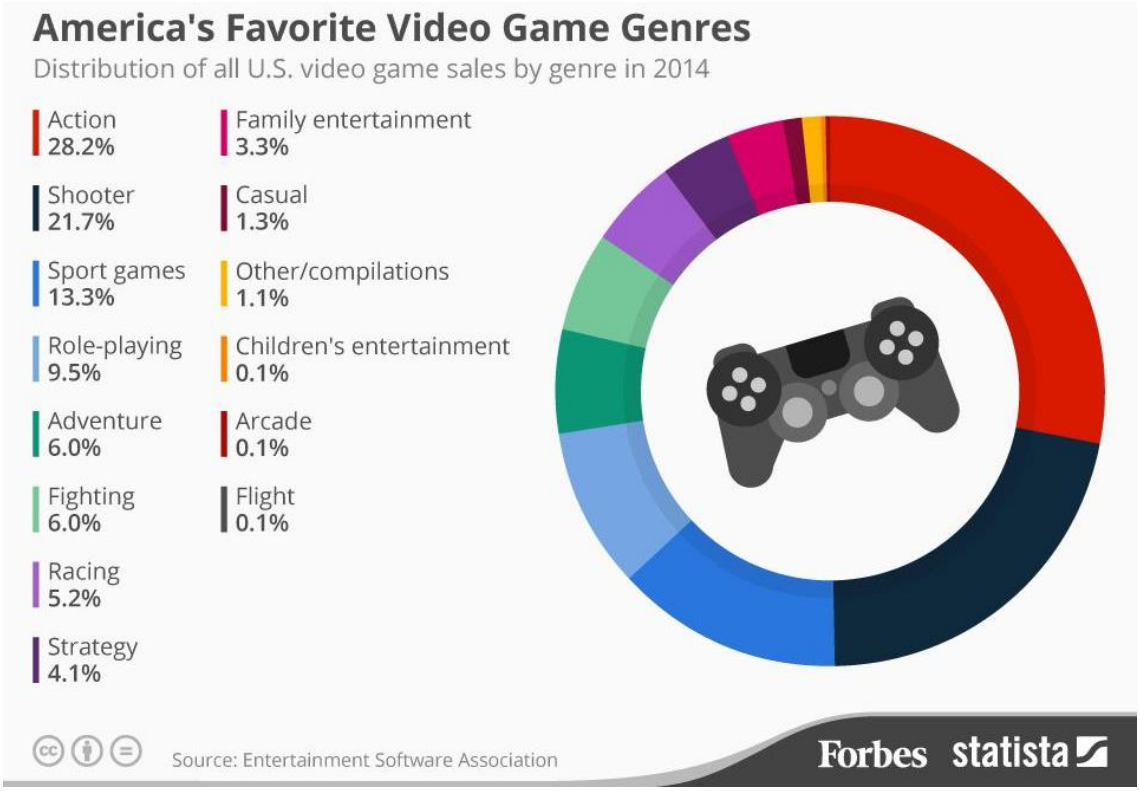

Figura 2.8: Géneros más vendidas en 2014 en EE. UU.

2.3 Serious Game

Paralelamente a los géneros mencionados, se fueron desarrollando los *Serious Game* o 'juegos serios'. Estos no son muy conocidos debido a que su principal objetivo no era, ni es, la comercialización global. Su propósito fundamental no es la pura diversión y depende del tipo de *Serious Game* del que se trate, ya que los hay orientados a gran variedad de ámbitos, como la defensa, la educación, la ingeniería o la sanidad.

Muchos de estos juegos consisten en la simulación de situaciones reales y tienen como objetivo entrenar a los futuros profesionales. Este tipo de juegos serios son muy comunes en el ambiente militar y en el de sanidad. En el primero se pueden crear simulaciones de vuelo o de disparo, en los que el soldado puede entrenarse sin requerir de los costes de realizar esas tareas realmente. En el segundo se suelen crear simulaciones de situaciones que pueden llegar a enfrentar los médicos, como operaciones o emergencias, aunque también se pueden usar para otros hábitos como la rehabilitación o la identificación de patologías. En ambos casos destaca por la seguridad con la que se pueden realizar las maniobras, ya que al no trabajarse en un entorno real permite a los inexpertos ganar experiencia sin poner en riesgo nada ni nadie [5].

Sin embargo, la mayoría de los *Serious Game* tienen como objetivo principal la pedagogía. No obstante, la diversión no se deja de lado e, incluso, se le intenta dar un aire competitivo (facilitado por tratarse de videojuegos). Incluso en las dos industrias anteriores hay juegos con este objetivo, como podría ser un juego de montar y desmontar armas o uno de operar a un paciente. Sin embargo, la pedagogía suele estar más enfocada a ámbitos como la educación, la ingeniería, la religión, la política, el manejo de emergencias y la planificación de ciudades.

La idea de usar los videojuegos con el objetivo de educar surgió en los 80 y se ha ido desarrollando hasta hoy en día, donde este tipo de juego cada vez tiene más importancia debido al avance de la tecnología y sus consecuentes beneficios. Entre las tecnologías que han permitido avances destacan los nuevos tipos de controladores existentes. Por ejemplo, los nuevos sensores de profundidad, como el *Kinect*, permiten rastrear la posición de la mayoría de las articulaciones del usuario, lo que podría permitir un seguimiento preciso de la movilidad de un paciente. Otro ejemplo podría ser la realidad virtual, la cual permite interactuar con los usuarios de forma más directa y similar al mundo real. Esta tiene la capacidad de mostrar a los niños animales salvajes o planetas lejanos de una forma mucho más visual que en los libros.

En cuanto a las ventajas que ofrecen este tipo de juegos es necesario destacar el costo. Este es desmesuradamente inferior al de simuladores tradicionales. Esto se debe a que no requieren de hardware especializado, tienen un público mayor, por lo que el precio por unidad tiende a disminuir, y puesto que se basan en situaciones reales ya tienen las bases bien asentadas y no hay que invertir tanto en creatividad e innovación.

Si hablamos de efectividad, hay que señalar que según un estudio realizado por la investigadora *Traci Sitzmann* en la Universidad de Colorado, los estudiantes que realizaron la formación mediante videojuegos obtuvieron de media mejores resultados. Siendo estos 11% superiores en conocimientos sobre hechos, 14% es conocimientos

basados en habilidades y 9% en niveles de retención de conocimiento. Además, otros estudios en años posteriores confirmaron los resultados. Se puede concluir entonces con que el aprendizaje mediante los videojuegos es efectivo [6].

Por último, se van a examinar algunos de los *Serious Game* existentes en el momento:

- *Wonder City (G4C, 2013).* Desarrollado por la cineasta *Kristy Guevara-Flanagan*, su temática trata sobre la visualización de las relaciones entre poder y género a través de la cultura popular. Es una aventura gráfica en la que se trata de concienciar al público mediante la historia sobre un tema específico.
- *Foad Force (Deepend & Playerthree, 2005).* Este juego simula las tareas diarias que llevan a cabo los voluntarios de la ONU para aliviar el hambre y la sed. Además, también trata de ciertos problemas médicos, de los refugiados y, en general, de las distintas situaciones que allí se viven. De esta forma, se quiere concienciar al jugador de la labor humanitaria que realizan [7].
- *Hackend (Incibe, 2016*). Es un videojuego gratuito destinado a enseñar ciberseguridad en las empresas. Se trata de una aventura gráfica, en la que se han de resolver nueve casos relacionados con la ciberseguridad de tu empresa virtual.
- *The Reach Game & The balance Game*. (Jaime Herreros Diaz, 2016). Este juego fue desarrollado por un compañero del equipo como TFG. Su función es la de ayudar en la rehabilitación de pacientes con discapacidad motora. El primero de los juegos está centrado en la movilidad de los brazos, ya que consiste en agarrar un objeto, moverlo y soltarlo. El segundo está basado en el equilibrio y la movilidad del paciente, puesto que consiste en esquivar objetos que vienen de frente. El avatar en ambos casos reacciona conforme el jugador lo haga, gracias al sensor de movimiento.

3 Usuario objetivo

En el siguiente punto se analizarán los usuarios a los cuales irá dirigido el juego. Ya que el proyecto tiene como finalidad ayudar a la rehabilitación motora, es obvio que será un S*erious Game* orientado a la salud, es decir, que va dirigido a pacientes que sufren alguna deficiencia motora.

Esta puede provenir de diferentes enfermedades o problemas. Los más comunes son los accidentes de tráfico, los ictus, las lesiones medulares, las ataxias y la esclerosis múltiple. Cada una de estas fuentes tienen una serie de síntomas y tratamientos distintos, por lo que nos vamos a centrar en un tipo de causa específica.

Hay que tener en cuenta que el proyecto realizado el año pasado, por Jaime Herreros Díaz, en este mismo grupo de trabajo, trató de la rehabilitación tras haber sufrido un ictus. Por tanto, se cree conveniente continuar en esta línea de trabajo. Debido a que ya se ha trabajado en la rehabilitación en sí misma (movilidad, progresiones…), este proyecto se centrará en la escala *Fugl-Meyer*, la cual es específica del ictus y sirve para valorar la severidad del déficit, planificar un tratamiento rehabilitador y realizar un seguimiento de la evolución del paciente tras el ictus.

Se comenzará analizando la patología a tratar, los tipos que existen, las causas, los factores de riesgo, el tratamiento y finalmente la rehabilitación. Se hará hincapié en esta última, puesto que en ella se basará el S*erious Game*.

3.1 Ictus

El termino ictus hace referencia al trastorno brusco del flujo sanguíneo cerebral que altera de forma transitoria o permanente la función de una determinada región cerebral. También es conocida como una enfermedad cerebrovascular (ECV), puesto que según la OMS (Organización Mundial de la Salud), esta es el desarrollo rápido de signos clínicos de disturbios de la función cerebral o global, con síntomas que persisten 24 horas o más, o que llevan a la muerte con ninguna otra causa evidente que el origen vascular [8].

Existen numerosas formas de clasificar los diversos tipos de ictus dependiendo de criterios clínicos, diagnósticos, pronósticos… Sin embargo, el método más común es según el mecanismo de producción, originándose así dos grandes grupos:

- **Ictus hemorrágico**. Suponen entre un 15% y un 20% del total. Este tipo de ictus se caracteriza por la rotura de una arteria y el consecuente exceso de sangre. Esto no sólo provoca la desnutrición de las células del cerebro, sino que también puede provocar un aumento de presión en él. La arteria puede ser intracerebral o de la superficie del cerebro. En el primer caso, el motivo de la rotura suele ser por hipertensión, aunque también se pueden dar malformaciones en los vasos

sanguíneos. En el segundo caso, lo más frecuente es un traumatismo craneal, provocado por un golpe o una caída.

- **Ictus isquémico**. Este tipo supone entre un 80% y un 85% de los casos. Es provocado por la isquemia cerebral, que consiste en la falta de oxígeno y nutrientes en las células, lo que les provoca lesiones. Esta carencia se produce cuando ocurre el taponamiento de una arteria, lo que impide que la sangre fluya. Si el periodo de escasez es lo suficientemente extenso, se ocasiona un infarto cerebral, es decir, el tejido muere. El taponamiento puede producirse por un coágulo, un tumor que comprima la arteria o un trombo ("coágulo de sangre que se forma en el interior de un vaso sanguíneo"). Por último, destacar que, si no se produce infarto cerebral, el ataque cerebral es llamado 'Ataque isquémico Transitorio'.

Hay que resaltar que, en el caso de los isquémicos, si el tiempo de ausencia de sangre ha sido bajo, los efectos pueden desaparecer instantáneamente. Aun así, deber ser valorados por un neurólogo, puesto que se podría ver afectadas ciertas zonas del cerebro. También es importante resaltar que los hemorrágicos suelen ser más graves, ya que presentan un mayor riesgo de mortalidad, pero son menos graves a medio plazo.

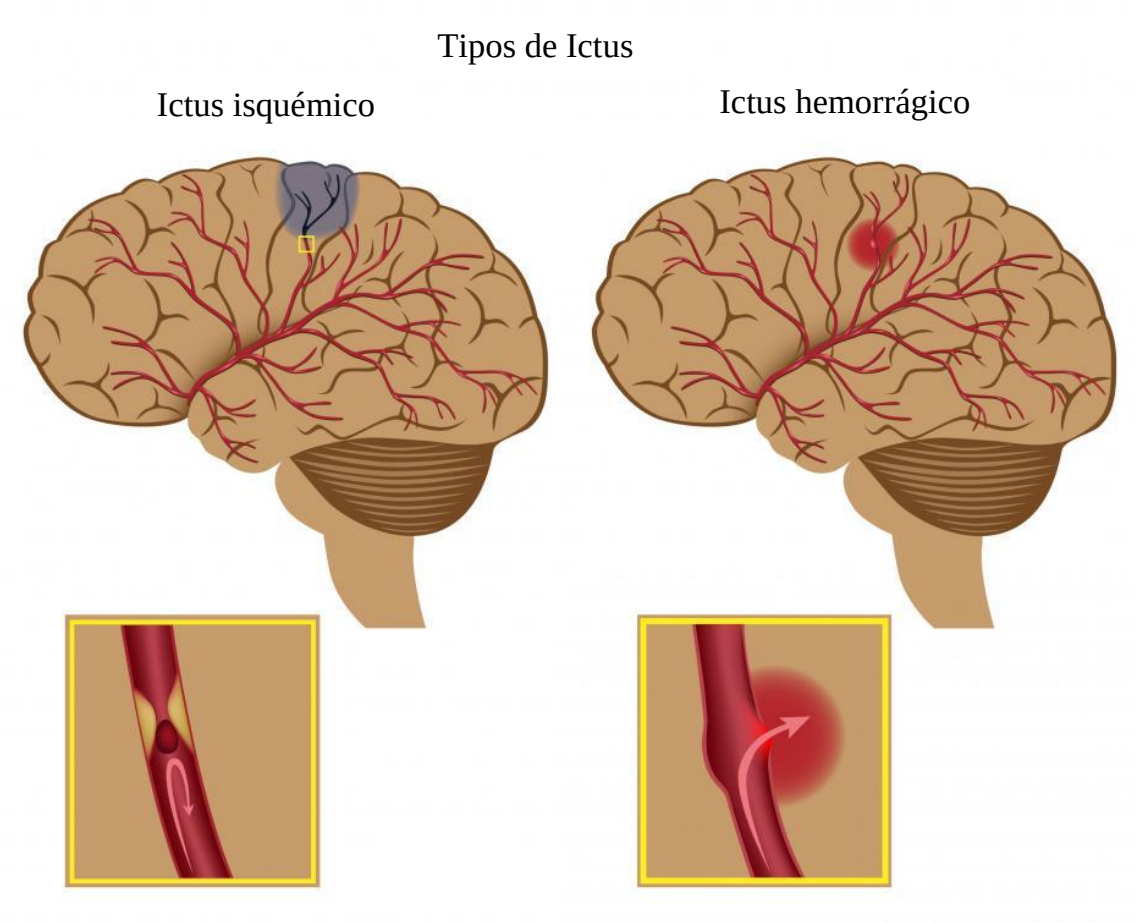

Figura 3.1: Tipos de ictus

El ictus es considerado como una de las principales causas de muerte en los países desarrollados. Según la OMS, un 10% de las muertes en países desarrollados se deben a la ECV. Además, el ictus es primer motivo de incapacidad permanente, tanto física como intelectualmente, y el segundo motivo de demencia. Por tanto, supone una importante carga social, sanitaria y económica. Se calcula que en los países desarrollados entre el 3 y el 4% del gasto sanitario es debido al ictus. Además, hay que tener en cuenta que se mantiene estable debido a que surge con la edad, y, puesto que actualmente la población de los países desarrollados está envejeciendo, es esperable que en los próximos años la cantidad de sujetos con esta patología aumente [8].

Una vez instaurado el ictus, es preciso mantener una atención sanitaria dirigida a la rehabilitación y a la prevención, debiéndose mantener esta última de forma indefinida. Esto puede llegar a consumir una gran cantidad de recursos, debido al requerimiento de profesionales. Aquí es donde el proyecto tiene su lugar. Se pretende automatizar todo lo posible esta rehabilitación, para que consuma los menos recursos posibles, algo necesario teniendo en cuenta la tendencia actual.

3.1.1 Factores de riesgo

Un factor de riesgo, según la OMS, es cualquier rasgo, característica o exposición de un individuo que aumente su probabilidad de sufrir una enfermedad o lesión, respecto al resto de la población. La identificación de estos permite establecer estrategias y medidas de control en los sujetos que aún no han sufrido la enfermedad, mientras que en los que ya la han sufrido, permite prevenir y reducir las recidivas (“reaparición de una enfermedad poco después del periodo de convalecencia”).

La probabilidad de sufrir un ictus aumenta exponencialmente dependiendo de los hábitos de riesgo que se tenga. Además, en la mayoría de los casos se producen secuelas permanentes, por lo que la mejor forma de abordar el problema es la prevención, lo que exige la identificación y el control de los factores de riesgo.

El factor de riesgo más destacable es la hipertensión arterial, cuyo tratamiento puede reducir el riesgo de ictus hasta en un 42%. También destaca la edad, ya que, a partir de los 55 años, la incidencia de ictus aumenta más del doble en cada década. El tabaquismo es otro de los más importantes, los fumadores tienen tres veces más posibilidades de sufrir un ictus que los no fumadores. Otros factores de riesgo son la diabetes, la dislipemia, la obesidad, el sedentarismo, la elevación del fibrinógeno, el alcohol, las drogas o las enfermedades cardiacas.

Estos, además, se pueden clasificar en modificables y no modificables, tal y como se puede apreciar en la **tabla 3.1**, donde se muestra la lista completa de factores de riesgo.

MODIFICABLES	NO MODIFICABLES
Hipertensión arterial	Edad
Cardiopatía (FA, IAM, miocardiopatías)	Sexo
Tabaquismo	Factores hereditarios
Anemia de células falciformes	Raza/etnia
AITs previos	Localización geográfica
Estenosis carotídea asintomática	Estación del año
Diabetes Mellitus	Clima
Homocisteína	
Hipertrofia ventricular	
Dislipemias	
Anticonceptivos orales (>6 años)	
Consumo excesivo de alcohol	
Consumo de drogas (cocaína, anfetaminas)	
Sedentarismo	
Obesidad (fundamentalmente troncular)	
Factores dietéticos (sal, grasas saturadas...)	
Hematocrito elevado	
Hiperinsulinemia/resistencia a la insulina	

Tabla 3.1: Factores de riesgo en el ictus [8].

3.1.2 Síntomas y diagnóstico

Hay que destacar la importancia de saber los síntomas que preceden a un ictus, ya que numerosos estudios han demostrado que cuanto antes se llega al hospital, menores son las secuelas. Por ello, la población general debería conocer los síntomas de alarma que preceden a este tipo de ataques, es especial, aquellos individuos cuyos factores de riesgo sean mayores.

También hay que recalcar que son indoloros, por lo que la ausencia de dolor no debe retrasar la petición de ayuda. En especial si la aparición de los síntomas es repentina y de corta duración, ya que podría tratarse de un Ataque Isquémico Transitorio, el cual podría ser la anticipación de un ataque más grave en los próximos días o semanas.

Los principales síntomas del ictus son:

- Pérdida de fuerza o adormecimiento repentino en la cara o en uno de los lados del cuerpo (hemidistonía).
- Incapacidad para hablar correctamente, siendo difícil expresarse de forma que los demás lo entiendan.
- Pérdida repentina de visión, parcial o total, en uno de los ojos.

Estos síntomas pueden venir acompañados de otros como vértigos, inestabilidad, caídas bruscas, cefalea acompañada de vómitos o incluso una disminución brusca del nivel de conciencia [9].

Si se sufren estos síntomas o se cree haberlos sufrido hay que ir al médico para que este diagnostique si se trata de un ictus o no. Los síntomas que presente el paciente pueden ser suficientes para indicar al neurólogo la posibilidad de que se trate de un ictus. Actualmente, las pruebas de imagen que se utilizan para diagnosticarlo son el TAC o el escáner, la resonancia magnética y la ecografía-doppler carotídea o transcraneal. Una exploración física también puede ser útil para valorar que área del cerebro se ha visto afectada.

El TAC sirve para saber el tipo de ictus que se ha sufrido, además permite visualizar la extensión del mismo y descartar otras enfermedades con síntomas similares. La resonancia define las zonas afectadas para poder así programar un tratamiento adecuado a cada paciente. Por último, la ecografía de las arterias carótidas permite detectar posibles trombos y el Doppler transcraneal valorar el estado de las arterias en el territorio cerebral afectado, fundamental para conocer la gravedad y el pronóstico [9].

3.1.3 Secuelas y tratamiento

A raíz de haber sufrido un ataque de ictus, suelen surgir ciertas secuelas y complicaciones, tanto físicas como psicológicas. Las más comunes son las alteraciones del estado de ánimo, concretamente se tiende a ser más depresivo, lo cual ralentiza la rehabilitación. Los sujetos afectados también suelen ser más irritables, sufren de ansiedad y es común la desmotivación.

En cuanto a las secuelas físicas más destacables, estas son [4]:

- Afectación al movimiento. Es posible la pérdida parcial o total de fuerza es el lado afectado. También puede haber problemas de control y coordinación.
- Trastorno en el lenguaje. Puede darse el caso en el que el paciente no sea capaz de comprender el lenguaje, tenga ciertas dificultades al hablar o incluso no pueda emitir palabra alguna.
- Trastorno en la sensibilidad. El lado afectado puede dejar de ser sensible al tacto, o sentir hormigueos.
- Propensión a las caídas. Puede haber una pérdida de habilidad motora, que junto a la debilidad muscular desemboca en una caída.
- Dolores. Pueden darse en el lado afectado o superficialmente. En ambos casos, se tratan con analgésicos.
- Espasticidad. Consiste en la contracción permanente de un músculo, llegando a ocasionar dolor.
- Hemianopsia. Se trata de la pérdida parcial de la visión, concretamente del campo visión de la zona afectada.

Por todo lo mencionado, es evidente que se debe aplicar un tratamiento adecuado, de forma que estas secuelas puedan reducirse al mínimo. Por lo general, se dice que para un buen pronóstico es necesario que los ictus sean tratados en menos de tres horas por un equipo de neurólogos.

Dependiendo del tipo de ictus con el que se encuentren los médicos estos realizaran un tratamiento u otro. Normalmente, si se trata de un ictus isquémico, suelen tratar de inyectar medicamentos que disuelvan el coágulo o el trombo, de forma que se pueda recuperar el flujo sanguíneo. El fármaco más usado para conseguirlo es la aspirina o ácido acetilsalicílico.

En los últimos años, en los centros especializados se ha empezado a realizar la trombolisis inyectando un medicamento, que no solo disuelve el coágulo, sino que también consigue la restauración de las funciones neurológicas pérdidas durante el ictus. Sin embargo, este método tiene un gran riesgo de sangrado, por lo que es imprescindible que los médicos seleccionen a los pacientes que realmente se van a ver beneficiados en función de la edad, el tiempo de evolución del ictus, los antecedentes…

Si hay una hemorragia, el tratamiento se vuelve más complejo. Las formas más comunes de intervenir son mediante una angiografía intervencionista o mediante un cateterismo para reparar la arteria dañada colocando *stents*. Un *stents* es un pequeño tubo de malla de metal que se expande dentro de una arteria permitiendo el paso de la sangre por el interior. También se pueden insertar clips o grapas en los aneurismas que causen el sangrado. Si la vida del paciente corre peligro por la localización o la extensión de la hemorragia, se puede recurrir al drenaje quirúrgico, mediante la introducción de un catéter para sacar la sangre al exterior [9].

Una vez que el tratamiento inmediato se ha finalizado es preciso comenzar a pensar en la rehabilitación. No solo para la recuperación temprana, sino también para evitar futuros problemas. En los tres meses siguientes a la aparición del ictus, la posibilidad de que se repita es considerable, y durante el primer año no baja de un 14%. No hay que olvidar que se trata de una enfermedad grave e incapacitante. Durante el primer mes tras sufrirlo, la mortandad oscila entre un 10 y un 20% [8]. Además, las complicaciones que vienen de su lado pueden perjudicar la vida del paciente y su entorno de por vida. Por ello, es preciso realizar la mejor rehabilitación posible.

3.1.4 Rehabilitación

Hay que recalcar que la mayoría de las complicaciones inmediatas tras el ictus están relacionadas con la inmovilidad, por lo que es recomendable favorecer la movilidad del paciente tan pronto como sea posible. Es más, se considera que el inicio de la rehabilitación durante las primeras 24-72 horas tras el ictus es beneficioso. Además, se ha comprobado que los pacientes que inician su tratamiento rehabilitador en la primera semana después del ictus tienen un grado de discapacidad menor y una mayor calidad de vida a largo plazo. Sin embargo, no es suficiente con un inicio precoz en la rehabilitación, otros factores como la continuidad, la duración, la intensidad y el soporte sociofamiliar adecuado también son fundamentales para que esta sea óptima.

Es necesario aclarar que el objetivo principal de la rehabilitación es ayudar al paciente a adaptarse a sus déficits, a compensarlos, no a eliminarlos. Esto se debe a que,

en la mayoría de los casos, la lesión se recupera total o parcialmente en un tiempo variable, pero dependiendo de la gravedad del ictus puede que nunca se llegue a recuperar.

El proceso de rehabilitación es complejo, requiere abordar las deficiencias motoras, sensoriales y neuropsicológicas desde múltiples ángulos, usando para ello técnicas de reeducación que se basan en la plasticidad neuronal y en la rehabilitación orientada a tareas. Es importante que el paciente y la familia se impliquen activamente.

Los mayores progresos se aprecian en los dos primeros trimestres de la rehabilitación, aunque hay casos en los que las mejoras se prolongan hasta los dos años después de haberse producido el ictus. Las primeras funciones en recuperarse son las relacionadas con los movimientos voluntarios, mientras que la restauración del habla o del equilibrio suelen venir con el tiempo.

Los programas de rehabilitación que se usan actualmente surgieron alrededor de los años 80 y consisten en determinadas técnicas de terapia física, ocupacional y de logopedia. En el terreno de la recuperación física, la terapia se basa en cinco puntos: instrucciones verbales simples, demostraciones visuales, guía manual, *feedback* positivo y repetitividad. Las más importantes consisten en: marchar sobre una cinta con una suspensión parcial del peso corporal, producir un movimiento inducido mediante la restricción del lado sano, estimulación sensitivomotora asistida con robots, fortalecimiento muscular y reacondicionamiento físico.

Para finalizar, destacar que hay distintos tipos de unidades encargadas de la rehabilitación del ictus. Estas son las unidades de rehabilitación en Hospital de agudos y UI (Unidades de Ictus), las unidades de rehabilitación ambulatoria hospitalaria y las unidades de rehabilitación domiciliaria. Todas estas unidades están formadas por equipos multidisciplinarios de expertos en el tema y tienen funciones distintas. La primera tiene como objetivo asegurar el control postural correcto, evitar la inmovilidad y mantener una función respiratoria y deglutoria correcta. La segunda entra en acción cuando el paciente ya está en buenas condiciones, se encarga de realizar una rehabilitación continuada en el tiempo. Por último, la tercera es la encargada de desplazarse al lugar de residencia del paciente, si esto fuera preciso [8].

3.2 Fugl-Meyer

Es indiscutible que la rehabilitación es un proceso fundamental para disminuir las secuelas de un ictus, pero, como se sabe si esta se está realizando correctamente, o si se ha logrado alguna mejoría. La respuesta es simple, de la misma forma en la que se conoce la gravedad del estado de un paciente o la magnitud de la secuela, mediante una escala de valoración.

Por tanto, se puede decir que una escala de valoración del ictus es una herramienta usada por los médicos neurorrehabilitadores para cuantificar de la forma más sencilla posible la extensión de un ictus en un momento en el tiempo. Esto permite, no solo saber el estado del paciente con exactitud, sino también facilitar la transmisión de información y resultados.

Existen una gran variedad de escalas diseñadas específicamente para el ictus. Esto se debe a que cada una tiene en cuenta ciertos parámetros o usa determinados instrumentos, lo que provoca que estén enfocadas más en unos problemas que en otros. Por tanto, dependiendo de lo que se busque, hay que usar una escala específica. Los distintos tipos de escalas según su función son: escalas del déficit del funcionamiento, escalas de limitación de la actividad y escalas de restricción de la participación.

En nuestro caso, la única que podríamos tratar es la de funcionamiento, por lo que nos centraremos en ella. Según Begoña M.ª Ferrer González [8], las más usadas de este tipo son:

La *National Institute of Health Stroke Scale (NIHSS*). Es la más empleada en la valoración de la severidad del ictus. Está constituida por 11 ítems que permiten detectar mejoría o empeoramiento neurológico. Por tanto, se considera que puede predecir el resultado del ictus a largo plazo.

La escala canadiense. Es una escala sencilla, que se centra en los aspectos cognitivos como la consciencia, el lenguaje o la orientación. Permite conocer las posibilidades de comunicación con el paciente.

La escala modificada de *Ashworth*. Es una graduación que mide la espasticidad muscular. Lo consigue gracias a la resistencia que opone el músculo al movimiento pasivo. El médico mueve las articulaciones del paciente y, en función de la resistencia ofrecida, lo valora del 1 al 5.

El *Minimental State Examination* (mini-examen cognoscitivo). Como su propio nombre indica, es un test cognitivo que se utiliza para la evaluación de posibles demencias. Los ítems están agrupados en 5 apartados y analizan la orientación, la memoria de fijación, la concentración, el lenguaje y el recuerdo diferido.

La escala *Fugl-Meyer*. Es una de las medidas de la funcionalidad del cuerpo tras el ictus más ampliamente reconocidas y clínicamente relevantes. Fue creada en 1975 por *Axel R. Fugl-Meyer, Lisbeth Jääsco, Ingerged Leyman, Sigyn Olsson y Solveig Steglind* en Suecia. Consta de 113 ítems divididos en 5 grupos. Cada uno de estos cubre ciertas características del estado del paciente. Las tres grandes partes que se evalúan son la

función motora y el equilibrio, la sensibilidad y el rango de movilidad pasiva, junto con la aparición de dolor. Cada uno de los ítems tiene que ser puntuado por el examinador, pudiéndose obtener 3 resultados distintos, 0, 1 y 2. El resultado, por tanto, se da en una escala numérica, correspondiéndose una mayor puntuación con un mejor estado del paciente.

En la **tabla 3.2** se muestra una simplificación de lo que sería la prueba. Como se puede observar, la puntuación máxima coincide con el doble de los ítems existentes, lo que equivaldría a obtener 2 puntos en todos. También se pueden apreciar los 5 grupos, separados mediante una línea azul más gruesa, y las tres dimensiones en las que se trabaja.

DIMENSIÓN	DOMINIO	NÚMERO DE ITEMS	Puntuación mínima y máxima	Puntuación total
FUNCIÓN MOTORA Y EQUILIBRIO	Función motora en el miembro superior	33	0-66	114
	Función motora en el miembro inferior	17	0-34	
	Equilibrio	7	0-14	
SENSIBILIDAD	Sensibilidad estésica	4	0-8	24
	Sensibilidad propioceptiva	8	0-16	
RANGO DE MOVILIDAD PASIVA Y DOLOR ARTICULAR	Rango de movilidad articular pasiva	22	0-44	88
	Dolor a la movilidad pasiva articular	22	0-44	
TOTAL		113		226

Tabla 3.2: Simplificación del test *Fugl-Meyer* [8].

Su aplicación completa es algo compleja, pero si se hace puede servir para valorar la funcionalidad del ictus. No obstante, también se puede valorar los diferentes apartados globales de forma independiente, pudiéndose así clasificar como una escala que evalúa el déficit motor. Esto permite medir de forma objetiva la recuperación motora y sensorial del paciente afectado por el ictus. De esta forma, si un médico solo quiere evaluar uno de los dominios en los que se divide el test, puede hacerlo, sin necesidad de realizar el resto de pruebas. Con este mismo objetivo se han desarrollado diferentes versiones del test especializadas en ciertos propósitos.

Tras analizar las características de esta escala, se ha llegado a la conclusión de que sería apta para automatizarla, al menos, en parte. Dominios como la sensibilidad o la movilidad pasiva no se podrían cuantificar, sin embargo, otros como la función motora si. Y, puesto que como ya se ha explicado, no es necesaria la totalidad del test para que sea válida, se considera que la automatización de la escala *Fugl-Meyer* cumple con los requisitos del proyecto. Por tanto, se analizarán los ítems que puedan realizarse sin riesgos para el paciente y se diseñará el S*erious Game* en función de estos (tanto el análisis como

la elección de los ítems a implementar se realizará en el apartado 5). De esta forma el juego podrá evaluar la evolución del paciente en el ámbito de la función motora.

Sin embargo, esto plantea un debate, realmente es la realidad virtual en la rehabilitación una buena alternativa a la rehabilitación tradicional. Esta última, claramente, ayuda a mejorar las funciones motoras tras un ictus. Sin embargo, presenta una serie de limitaciones: consume una gran cantidad de tiempo a los especialistas, además de recursos, es una labor muy intensa, depende de la relación con el paciente, es repetitivo y sistemático, requiere el transporte a instituciones especializadas y el coste es considerablemente alto.

Por todo ello, la realidad virtual se está convirtiendo en una alternativa a tener en cuenta. Esta soluciona la mayoría de los problemas anteriormente nombrados. Además, proporciona ventajas que la tradicional no podría aportar, como, por ejemplo, hacer que la tarea deje de ser repetitiva, puesto que se añade la diversión de un videojuego. Esto hace que el paciente no pierda la motivación. También disminuye la labor del terapeuta; es portable, por lo que no haría falta asistir a centros especializados; permite analizar los movimientos de forma precisa, en tiempo real y con un fácil acceso y lo más importante, su costo es mínimo en comparación a otros métodos, lo que hace que sea una tecnología accesible para cualquier centro rehabilitador que la precise.

En cuanto a los métodos de este tipo usados en los últimos años se pueden destacar entrenamientos en cinta con realidad virtual, entrenamientos con sistema de tracking y entrenamiento a través de *Serious Game*. El primero es, a grandes rasgos, un simulador de paseos, en el que el paciente caminaba en una cinta y la realidad virtual hacía que el entorno fuese un parque, una ciudad… los resultados obtenidos muestran que no hay ventajas significativas al usar este método. El segundo consiste en medir los movimientos del paciente mediante distintos sensores, mientras se le ponen varios objetivos. En este caso, si que se suelen lograr mejoras significativas. Por último, el uso de *Serious Game* demostró beneficios en el paciente, siempre y cuando el juego estuviese específicamente diseñado para su patología. De esta forma, el *feedback* recibido es positivo, ya que, al tener deficiencias motoras, el paciente no puede optar a conseguir objetivos viables para el público general.

Para concluir, destacar que este proyecto será un S*erious Game* que hará uso de sistemas de *tracking* para poder así monitorizar el progreso del paciente. Con esto se pretende maximizar el rendimiento y la satisfacción del paciente, puesto que contaría con los beneficios de ambos métodos.

4 Tecnologías

Una vez llegado a este punto, explicado el objetivo del proyecto y la patología a tratar, es hora de centrarse en la tecnología existente en el mercado, con la cual se podría llevar a cabo el proyecto. En este apartado se analizarán las más importantes y se compararán entre ellas. De este modo, se podrá elegir la combinación óptima para el desarrollo del *Serious Game*.

Para ello, en primer lugar, hay que preguntarse: ¿cómo se hace un juego?, ¿cuánto tiempo puede necesitar y de cuánto tiempo se dispone?, ¿cuál es el método más adecuado en nuestra situación? En los siguientes apartados se responderán a estas preguntas, exponiendo los pros y los contras de cada método. Además, se comentarán los dispositivos de captura de movimiento más importantes actualmente. Tras esto, se realizará una comparativa entre las distintas tecnologías de los diferentes campos, pudiendo así elegir la mejor opción justificadamente.

4.1 Herramientas de desarrollo

Un videojuego puede crearse con una infinidad de métodos distintos, dependiendo de factores como el presupuesto, el tiempo destinado, la calidad buscada etc. Elegir uno de estos métodos es parte de las primeras etapas en la elaboración del juego, junto a la búsqueda de un editor, el desarrollo de un guion o la concepción de las mecánicas del juego, pasos que se obviarán por no tener presentes en el proyecto.

Todos estos métodos se basan de una forma u otra en herramientas de desarrollo, las cuáles consisten en programas o aplicaciones que tienen cierta importancia a la hora de crear, gestionar o mantener un programa.

Estos métodos requieren una aplicación de conocimientos muy amplia, tales como programación, sistemas operativos, inteligencia artificial, gráficos por ordenador o ingeniería de software. Todo esto, está fuertemente relacionado con la tecnología de la que se dispone, fundamental para el desarrollo de videojuegos, y la producción de herramientas de desarrollo [10].

Sin embargo, la producción de estas es una tarea extraordinariamente compleja, por lo que según ha ido evolucionando el mercado, se ha ido dejando de lado la exclusividad por la productividad. En la actualidad muy pocas empresas desarrollan la totalidad de sus herramientas, la mayoría compran licencias para usar programas de terceros en la creación de sus videojuegos. Eso crea una situación en la que muchas empresas están usando las mismas herramientas, lo que ha llevado a que estas deban tener una flexibilidad mucho mayor, con la cual se pueda lograr la diferenciación.

Aun así, hay empresas que siguen creando sus propias herramientas desde cero, lo cual incrementa significativamente los costes del proyecto. A cambio, se consiguen

unas características mucho más orientadas a su videojuego, lo que aumenta de forma significativa el rendimiento.

En cuanto a las distintas herramientas existentes en el mercado, se podría considerar como la más popular el uso de motores gráficos, los cuales se detallarán en el siguiente punto. Sin embargo, hay muchas más alternativas, como por ejemplo el desarrollo nativo usando el SDK (*Software Development Kit*) estándar para una plataforma o el uso de bibliotecas multiplataforma para lenguajes de propósito general como por ejemplo java y LWJGL (*Lightweight Java Game Library*) una librería de código abierto para el desarrollo de videojuegos [11].

Por lo tanto, hoy en día, gracias a la gran cantidad de opciones disponibles en el mercado, se pueden crear videojuegos independientemente del nivel de programación del que se disponga, algo inconcebible hace unos años atrás. Si no se tienen conocimientos previos de programación se pueden usar programas como *GameMaker*, enfocado a una interfaz gráfica y con la programación ya implementada de base. En el otro extremo, si se posee una gran experiencia programando, se pueden crear videojuegos desde cero, simplemente con la ayuda de un SDK o unas librerías. Este método tiene la ventaja de no estar sujeto a limitaciones de ningún tipo, ya que es el propio programador el que configura todo desde cero. Y, como última opción, existe la posibilidad de usar un motor gráfico, más complejo y flexible, en el cual es necesario programar, pero sigue disponiendo de las opciones gráficas que aumentan la productividad significativamente.

4.1.1 Adobe AIR

Adobe AIR (*Adobe Integrated Runtime*) es una herramienta con la cual se podrían desarrollar juegos desde cero. Consiste en un sistema de ejecución multiplataforma que permite el uso de *Flash*, *Action Script*, HTML *o Javascript* para la creación de aplicaciones RIA (*Rich Internet Applications*), las cuales se usan como aplicaciones de escritorio [12].

Esto tipo de implementación tiene un beneficio muy importante, el cual es que, al usarse como aplicación de escritorio, tiene acceso al sistema de archivos y al almacenamiento. Sin embargo, también tiene sus desventajas, como que requiere de empaquetamiento, permisos e instalación, mientras que una aplicación similar en un navegador no requiere de estos pasos, viéndose a cambio, más limitada en el almacenaje de datos.

Figura 4.1: Logo de *Adobe AIR*.

Es una herramienta multiplataforma, que permite el desarrollo de aplicaciones, juegos y videos tanto en ordenador como en dispositivos móviles. Concretamente, es operativo en *Windows, Mac OS, Android, iOS* y *BlackBerry Tablet OS*. Actualmente, hay tres entornos de desarrollo: *HTML/AJAX, Adobe Flash Builder 4 y Adobe Flash CS5*.

Al ser una herramienta destinada a hacer correr los juegos, no tienen interfaz por si misma. Esto quiere decir, que el software se desarrolla enfocado en Adobe AIR y no es un sistema operativo específico.

Para la creación de juegos, cuenta con una serie de herramientas y *frameworks* [12].

En cuanto a las herramientas destacan:

- *Gaming SDK*, que provee de los elementos necesarios como para monetizar un juego a través de las múltiples plataformas.
- *Flash Builder*. Ambiente de desarrollo especializado en el desarrollo de videojuegos.
- *Scout*. Orientado a mejorar el desarrollo de forma simple e intuitiva.

En cuanto a los *frameworks* se podría diferenciar los siguientes:

- *Native Extensions*. Permite trabajar de forma nativa, lo que deriva en conocer las especificaciones de la plataforma o sus compatibilidades.
- *Away3D*. Es un motor gráfico de código abierto, que permite el renderizado y la realización de cálculos con modelos en 3D.
- *CrossBridge*. Proyecto de código abierto que permite la combinación de C y C++ para correr el juego en *Flash Player* con aceleración de la GPU.

El costo de *Adobe AIR* es de cero, al igual que *Adobe AIR SDK*, el cual contiene todo lo necesario para desarrolladores no experimentados. Sin embargo, otras herramientas complementarias si que tienen un precio, dependiendo de múltiples factores, como el periodo por el que se va a contratar o los servicios exigidos.

4.1.2 GameMaker: Studio

Es una herramienta de creación de videojuegos centrada principalmente en juegos en 2D (2 dimensiones) o con vista isométrica, aunque también ofrece la posibilidad de crear gráficos y efectos en 3D (3 dimensiones). Caber recalcar que la funcionalidad de 3D viene limitada al apartado gráfico, no hay soporte oficial y debe realizarse mediante código, siendo este el mayor problema por el tipo de desarrollador que escoge este motor [14].

En su inicio, el nombre de este software era *Animo*. Fue lanzado en 1999 y en principio, el objetivo del proyecto era ayudar a los estudiantes con una herramienta de animación, pero finalmente, se derivó hacia una herramienta de creación de videojuegos. En 2012 se estrenó *GameMaker: Studio*, que proporciona soporte para desarrollar,

además de en *Windows* y *Mac*, en *iOS*, *Android*, *Ubuntu* y páginas web. Actualmente, se está desarrollando *GameMaker: Studio 2*, cuyas betas ya están disponibles en el mercado.

Figura 4.2: Logo de *GameMaker: Studio*.

Esta herramienta se puede adquirir de forma gratuita para probarla. Sin embargo, hay que pagar una serie de tasas, dependiendo de la plataforma en la que se quiera desarrollar. Estas tasas, junto con las características de cada licencia vienen detalladas en la **figura 4.3**.

	Trial	Desktop	Web	UWP	Mobile	PS4	Xbox One	Ultimate
Unlimited Resources		✓	✓	✓	✓	✓	✓	✓
Expert Features		✓	✓	✓	✓	✓	✓	✓
Target Platform(s)	Windows TEST only	Windows, Mac, Ubuntu	HTML5	Microsoft UWP	Android, iOS	PS4	Xbox One	All Platforms
GameMaker: Studio 1.4 Professional Access		✓	✓	✓	✓	✓	✓	✓
Marketplace		✓	✓	✓	✓	✓	✓	✓
Support	GMC	✓	✓	✓	✓	✓	✓	✓
License Type	Permanent	Permanent	Permanent	Permanent	Permanent	12 Month	12 Month	12 Month
	DOWNLOAD	$99.99	$149.99	$399.99	$399.99	$799.99	$799.99	$1500.00

Figura 4.3: Precios según la plataforma

Está basado en un lenguaje de programación interpretado y un kit de desarrollo de software (SDK) para desarrollar videojuegos, creado por el profesor Mark Overmars en el lenguaje de programación Delphi, y orientado a usuarios novatos o con pocas nociones de programación [15]. De ahí el gran problema de desarrollar en 3D. Para usuarios más avanzados existe GML o *Game Maker Language*, un lenguaje exclusivo de esta herramienta que permite extender sus características.

Usando la interfaz gráfica proporcionada, es posible crear un videojuego completo sin llegar a realizar una sola línea de código. A cambio de esto, el resultado será sencillo y poco personalizable, debido a las obvias limitaciones de un software que tiene que tener toda la programación necesaria hecha con anterioridad.

4.2 Motores gráficos

Un motor gráfico es un *framework* de *software* diseñado para la creación y el desarrollo de videojuegos. Es concebido con el objetivo de facilitar y acortar los periodos de desarrollo. Para esto, está dotado de una serie de funcionalidades: motor de renderizado para gráficos tanto en 2D como en 3D, motor de físicas capaz de detectar colisiones, respuesta a estas etc., animación, inteligencia artificial, escenario gráfico, audio, video, *scripting*, red, *streaming*, administración de memoria y otras muchas funciones más específicas que dependen de cada motor en concreto.

En los inicios de la industria de los videojuegos, todo el contenido del juego estaba en el ejecutable, algo inconcebible con el paso del tiempo, debido al aumento de complejidad en los nuevos juegos. Tras la aparición de los sistemas de ficheros, se empezó a separar el contenido en datos y ejecutable. Esto permitía modificar partes del juego sin acceder al ejecutable.

Sin embargo, no fue hasta el año 1993, con la aparición del *Doom Engine*, el establecimiento del término "*Game Engine*" o Motor Gráfico. El juego *Doom* fue diseñado con este motor, que por primera vez separaba el sistema de renderizado gráfico de la detección de colisiones, el audio, los objetos de juego y las reglas. Las ventajas de este sistema son evidentes, por lo que otros juegos, como *Quake* o *Unreal*, siguieron este método. Actualmente, lo más común son los motores de propósito general, es decir, motores que no se orientan a un juego en específico, sino que tienen como objetivo la creación de la mayor cantidad de juegos posibles, con las variaciones que esto implica. Por ello, el código es mínimo, genérico y reutilizable. También se suele ofrecer una amplia documentación, para así poder adaptarse a la complejidad dada [5].

Existen varios tipos de motores gráficos en el mercado. Según el tipo de licencia se pueden distinguir entre motores propietarios y motores *OpenSource*, permitiendo estos últimos el acceso al código fuente.

Si se distingue según el objetivo buscado hay tres tipos. Los más populares son los motores completos, como podría ser *Unreal Engine*. Estos proporcionan un resultado de máximo nivel, con la desventaja de que es necesario el scripting. Otro tipo son las llamadas herramientas de creación de videojuegos, como ya se ha comentado en el punto anterior (4.1). Por último, existen las librerías de apoyo que se encargan de facilitar el desarrollo. Algunos ejemplos son *DirectX y OpenGL*.

A continuación, se explicarán, de forma breve, las características principales de los motores gráficos:

- Gráficos. El motor gráfico es un procesador de geometría en tiempo real. Es imprescindible realizar optimizaciones en procesamiento y memoria. Para conseguir esto hay que analizar tres factores:
  - Determinación de la visibilidad. No es necesario representar los elementos no visibles de la escena.
  - Resolución. A menor resolución, mayor rendimiento. Se puede disminuir la cantidad de detalle en los objetos lejanos.

        - Geometría orgánica. Elementos en 3D complejos para los que hay que usar sistemas de partículas, modelos específicos (para nubes o agua) o software especializado como *SpeedTree*.
    - Edición de terrenos. Se utiliza para crear grandes extensiones de tierra. Es necesaria una distribución realista de estructuras u objetos.
    - Iluminación. Se encarga de proporcionar luz a la escena. Se tiene que tener en cuenta las interacciones entre objetos, es decir, los reflejos, las sombras… Estas últimas suponen un coste muy alto si se realizan en tiempo real.
    - Animaciones. Estas se pueden crear mediante ciertas herramientas, el motor debe de ser capaz de aplicarlas a los modelos necesarios.
    - Scripting. Sobresalen dos tipos. El scripting visual, en el cual no es necesario escribir código, ya que se hace mediante bloques. Y el lenguaje de alto nivel, que se usa para definir el comportamiento de las entidades del juego.
    - Motor de físicas. Destaca por un lado el sistema de simulación de física, cuya función es aplicar las leyes de Newton, calcular las fuerzas entre cuerpos, actualizar la posición y velocidad de los objetos etc. Y por el otro, el detector de colisiones, que se encarga de detectar el tipo de contacto entre modelados.

En cuanto a las ventajas frente a otras herramientas de desarrollo, sobresalen la portabilidad y la reusabilidad. La primera de estas destaca por el ahorro de ciclos de desarrollo completos por plataforma, puesto que se puede hacer un videojuego orientado a una plataforma y con unas pequeñas modificaciones lanzarlo en muchas otras. La segunda destaca por la posibilidad de usar el mismo software para la creación de múltiples videojuegos con la suficiente diferenciación entre ellos.

Además, mejora el flujo de trabajo, ya que cada programador puede trabajar con sus archivos y, posteriormente, unificarlos todos. También se disminuye el coste de producción. Esto se debe a que evitando producir un motor por videojuego, es decir, usando motores completos, se puede acortar los ciclos de desarrollo de forma significativa. En adición a esto, permite que el proyecto salga más rápido al mercado, favoreciendo el *time to market* [17].

La revista *Develop*, líder internacional para desarrolladores de juegos, realizó un estudio en 2014, en el que desarrolladores indies y expertos de la industria fueron preguntados por sus preferencias en cuanto a la mejor tecnología para la creación de sus juegos. En cuanto a motores gráficos, el resultado fue claro, situando a *Unreal Engine 4* a la cabeza, estando en segunda posición en el ranking global. El motor *Unity 5* fue situado a continuación, siendo la tercera posición global y terminando el top 3 se situó el *CryEngine*, que ocupa la onceava posición. Otros motores destacados son: el *GameMaker: Studio*, situado en la trigésimo quinta posición y el *Torque 2D/Torque 3D*, que de forma compartida poseen la septuagésima segunda plaza [18].

A continuación, se analizarán los motores con mejores resultados en esta encuesta. De esta forma, se determinará cuál es el más adecuado para el proyecto.

4.2.1 UNITY

Creado por Unity Technologies, este motor gráfico destaca por la variedad de plataformas en las que se pueden crear juegos. Esto lo consigue mediante una plataforma integral de desarrollo, desde la cual, se puede crear contenido interactivo, servicios de publicidad y análisis, soluciones colaborativas y otros elementos dirigidos para la plataforma deseada. Además, también destaca entre otros motores gráficos por su enfoque hacia los desarrolladores independientes que no tiene los recursos necesarios para costearse su propio motor, las licencias o las herramientas de desarrollo necesarias.

Figura 4.4: Logo de *Unity*.

En 2004, David Helgason, Nicholas Francis y Joachim Ante fundaron la empresa *Unity Technologies.* Su primer juego, *GooBall*, no tuvo éxito, pero todos reconocieron la calidad de sus bases técnicas, es decir, las herramientas de desarrollo usadas. Por ello, el 1 de junio de 2005 sacarían al mercado la primera versión de lo que hoy es un motor gráfico bien implementado en el mercado.

Esta primera versión se llamó Unity. Se lanzó en la Conferencia Mundial de Desarrolladores de Apple en 2005. El proyecto tuvo la tirada suficiente como para seguir trabajando en él. En 2008, con el 'boom' de iPhone, la empresa se convirtió en uno de los pocos desarrolladores bien afianzados a la plataforma, consiguiendo así, en ese periodo, un gran aumento de desarrolladores registrados.

En 2010 fue lanzado Unity 3, que impuso un punto de inflexión. En este momento, se introdujeron utilidades y herramientas en el programa enfocadas a captar el interés de los grandes estudios, mientras que se mantuvo el perfil llevado hasta el momento, en el que los pequeños equipos de desarrollo seguían teniendo las características necesarias para no tener que recurrir a un paquete poco asequible.

Desde ese momento hasta la actualidad, se ha seguido renovando pasando por las distintas versiones de Unity 4 y Unity 5, estando actualmente la versión 2017.1.1 presentada el 4 de septiembre de 2017 como la más novedosa.

Siguiendo la historia de la compañía se puede sacar en claro que su filosofía siempre ha sido la de apoyar a los desarrolladores independientes o que trabajen en pequeños equipos, sin recursos para crear un motor gráfico propio o adquirir licencias para uno. Tratando de hacer accesible la creación de videojuegos y contenido interactivo a la mayor cantidad de gente posible.

Esta política, entre otras, ha llevado a la compañía a ser el motor gráfico más usado en la actualidad. Según Unity Analytics, Unity tiene contacto con 770 millones de jugadores gracias a todos los juegos creados con su motor. Además, tienen una cuota de mercado del 45% tal y como se señala en la **figura 4.5**. Cabe remarcar que este dato se refiere a desarrolladores que usen su motor, no a porcentaje de juegos realizados con él [19].

Figura 4.5: Tasa de mercado según *Unity*.

En cuanto a leguajes de programación, soporta dos lenguajes de forma nativa. Por un lado, siendo el más usado y recomendable, se encuentra C#, un lenguaje estándar similar a Java o C++. Por el otro lado, se encuentra *UnityScript*, diseñado exclusivamente para este motor gráfico y modelado tras *JavaScript*. Cabe destacar que, en anteriores versiones, también se podían hacer *scripts* en Boo, pero debido a su ínfimo uso, a partir de la versión 5.0 se dejó de dar soporte a este lenguaje.

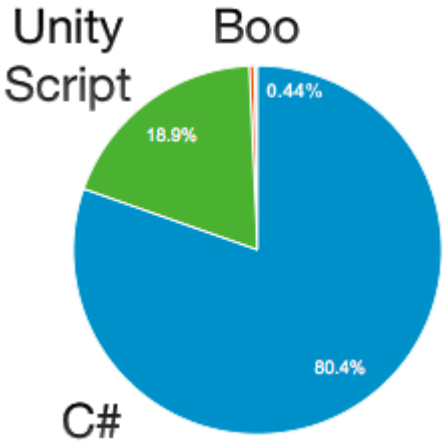

Figura 4.6: Lenguajes de programación en *Unity* a finales del 2014.

Fuente: *Unity Technologies.*

Actualmente, cuenta con tres licencias mediante las cuales obtienes el derecho de usar el motor. En primer lugar, se encuentra la versión de *Unity personal*, totalmente gratuita, destinada a estudiantes, aficionados y, en general, a cualquier principiante que quiera explorar la industria. Para obtener esta licencia es necesario registrarse en su página web. También se puede obtener *Unity Plus*, con un precio de 35$ por puesto y por mes, con su correspondiente mejora de servicio, entre las nuevas características que ofrece destaca el aumento de las plataformas en las que desarrollar. Por último, se

encuentra *Unity Pro*, con un costo de 125$ por puesto y por mes. En esta versión se encuentran disponibles todas las características del motor.

Cabe destacar que la versión a escoger viene delimitada por la política de empresa de Unity. Pudiéndose adquirir la versión Personal únicamente si los ingresos brutos anuales son inferiores a cien mil dólares, y la versión Plus si son inferiores a doscientos mil.

En la última versión de Unity, la llamada Unity 5.6, es posible desarrollar para un total de más de 25 plataformas, señaladas en la **figura 4.7**. Entre estas destacan Windows, Linux, IOS, Mac, Android, PS3, PS4, XBOX 360, XBOX ONE, Wii U y Switch. En adición, también se puede desarrollar para web y juegos de Realidad Virtual, también conocidos como VR (*Virtual Reality*) [16].

Figura 4.7: Plataformas de desarrollo en *Unity 5.6*

Además, cuenta con la *Asset Store*, una tienda virtual que permite el intercambio de bienes (*assets)* entre usuarios. Cuenta con un catálogo inmenso, dentro del cual hay desde modelos 3D, animaciones o audios, hasta texturas, partículas o incluso *script*s. Tiene el mismo funcionamiento que cualquier tienda virtual. Los usuarios, o incluso la propia empresa, suben contenido gratuito o de pago, al que los demás usuarios pueden acceder y adquirir. De esta forma se pueden crear juegos más rápido y de mayor calidad.

Por último, destacar que cuenta con una documentación amplia y detallada sobre el funcionamiento de todos los aspectos del motor, lo que, junto con la presencia de un foro técnico fuertemente transitado, hace que la solución de dudas se realice en plazos relativamente cortos. Esto ha provocado que se forme una gran comunidad alrededor, lo que garantiza, que al menos en el futuro próximo, se seguirá dando servicio a este motor.

4.2.2 Unreal Engine 4

Ultima versión del motor gráfico creado por *Epic Games Inc*. La primera versión de este motor se desarrolló en 1998, con la cual se hizo *Unreal*, uno de los FPS (*First Person Shooter*) mejor valorados de la historia. A partir de aquí, el motor fue evolucionando. En *Unreal Engine 2*, la empresa se centró en dar soporte a las principales plataformas del mercado, además, debido a las políticas de la empresa, se podía usar de forma gratuita sin fines lucrativos. Esto dio como resultado una gran cantidad de *mods* y proyectos universitarios creados con este motor.

En la siguiente versión, *Unreal Engine 3*, se dio un salto cualitativo en lo referido al apartado gráfico, tanto en el videojuego final como en la interfaz gráfica del

desarrollador, el cual podía crear escenarios sin prácticamente programar. Además, proporcionaba varias de las mejores herramientas y configuraciones de la industria, destacando por su potencia gráfica [20].

Actualmente, Unreal Engine 4 se ha convertido en uno de los motores más usados, principalmente por sus sencillas herramientas y sus acabados gráficos, los cuales pueden llegar a ser fotorrealistas. A continuación, se destacan varias de sus principales características [17]:

- *Scripting* visual mediante *Blueprints (Blueprint Visual Scripting*).
- Integra *middleware*: *OculusVR, Bink, Speedtree, Simplygon, Umbra, PhysX*.
- Compatible con *OpenGL* y *DirectX* 11 y 12.
- Imágenes de alto rango dinámico (HDR).
- *Anti-aliasing* temporal.

Figura 4.8: Logo de *Unreal Engine 4*

Algunos ejemplos de juegos famosos hechos con este motor son: *BioShock, Gears of War, SMITE, Mass Effect, Unreal Tournament o Tom Clancy's Rainbow Six*.

En la versión actual, *UnrealScript* ha dejado de ser el lenguaje de programación elegido, habiendo sido sustituido por C++. Su arquitectura es abierta, lo que permite la modificación del motor según las distintas necesidades y objetivos de los desarrolladores. Estas modificaciones se suelen realizar para lograr un mayor rendimiento en las distintas plataformas. Es compatible con una gran cantidad de estas, tal y como se puede ver en la **figura 4.9.**

Figura 4.9: Plataformas de desarrollo en *Unreal Engine 4*

En cuanto a la adquisición de la licencia, a partir de la última versión, esta es gratuita. Con la particularidad de que si se usa el motor con fines empresariales o

lucrativos *Epic games* recibirá un 5% de los ingresos brutos en royalties. Sin embargo, existe la opción de eliminar este porcentaje si se hace un contrato personalizado. Esta opción es negociable con la empresa por lo que existen infinidad de resultados [21].

Por último, resaltar que cuenta con una documentación accesible de forma gratuita, con una gran cantidad de contenido y lo suficientemente precisa como para que principiantes puedan considerar viable elegir este motor.

4.2.3 CryEngine V

Actualmente, es considerado uno de los motores más potentes de la industria. Su primera versión fue diseñada por la empresa alemana de desarrollo de software *Crytek*, para una demostración de características de los productos de *Nvidia*. Mostró un gran potencial, por lo que, posteriormente, fue usado como motor para crear *Far Cry*, un juego estrenado en 2004, que mezclaba el género FPS con el Sandbox. En 2006 todos los derechos fueron adquiridos por *Ubisoft*. Se han ido desarrollando nuevas versiones con el tiempo, la más novedosa es *CryEngine 5.4*.

Cabe remarcar que a partir de *CryEngine V* hubo un cambio de política en el modelo de licencias. Se introdujo el modelo de "paga cuanto quieras" para el uso y acceso al código fuente del motor. Como principales adiciones de esta versión, se permite *DirectX 12* de forma nativa, *Vulkan* y soporte para la realidad virtual [22].

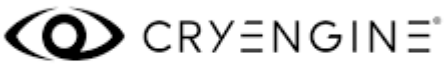

Figura 4.10: Logo de *CryEngine*.

Se ofrecen varios tipos de licencias según las necesidades del consumidor. El uso del motor es totalmente gratuito, mediante un modelo en el que se paga lo que se considere, estando exento de royalties. Sin embargo, también existen dos suscripciones mensuales: *Base Membership & Premium membership*, con un precio de 50 € y 150 € respectivamente. Estas opciones están dirigidas a desarrolladores que busquen un acceso opcional a entrenamiento adicional y soporte más allá del que proporciona la propia comunidad. Ambos paquetes permiten el acceso a un foro privado y tienen la opción de preguntar sus dudas a la propia empresa encargada del motor. Por último, existe una última opción llamada *Enterprise licensing*, la cual está dirigida a empresas de un tamaño mayor. Los términos de esta última son negociables y se puede ofertar paquetes de soporte extra [23].

Este motor no cuenta con una variedad de plataformas en las que desarrollar tan amplia como la de los anteriores. A pesar de eso, sigue contando con las más populares en el mercado: *Windows PC, Linux PC, PS4, Xbox One y Oculus Rift*.

CryEngine V usa dos lenguajes de programación, C++ y Lua. El primero es recomendable para las mecánicas clave que van a definir el juego, como el sistema de armas en un FPS o el sistema de diálogos en un RPG. El segundo es más efectivo para programar la inteligencia artificial (IA), los objetos o las misiones. En adición a estos, también se pueden usar otros lenguajes como C#, siempre que se cuenten con las librerías adecuadas.

Hay que resaltar la curva de aprendizaje, ya que en un principio puede ser un motor bastante complejo, lo que hace necesario un soporte acorde. Por ello, la propia empresa proporciona en su página web acceso a tutoriales, foros y a toda la documentación relacionada tanto con la interfaz, como con las bases del motor.

Para finalizar, resaltar varias de las características del motor, las cuales le hacen ser una de las mejores opciones en el mercado actual [24]:

- Sistema de partículas complejo, incluyendo sombras en humo volumétrico.
- SVOGI, el cual permite crear escenas fotorrealistas.
- Existencia de Marketplace, donde se puede comprar y vender assets.
- Reflejos locales en tiempo real.
- Anti-aliasing y Teselación de última generación.

4.3 Creación, modelado y animación de objetos.

En todo videojuego, hay un apartado gráfico y un apartado artístico. Ambos apartados se refieren a lo visual, al escenario, a los elementos que el jugador va a ver en escena. Sin embargo, el primero de estos conceptos está relacionado con la resolución de las texturas utilizadas, la cantidad de polígonos de los modelados, la calidad del sistema de partículas y, en general, con todo lo que depende de la potencia tecnológica de la que se disponga. En cambio, el segundo concepto deja lo técnico a un lado, centrándose en la puesta en escena, la ambientación o el diseño de los escenarios.

Si se unen ambos aspectos, se da lugar al apartado visual del juego. Este se puede crear de diferentes formas. Los motores gráficos de los que se ha hablado anteriormente tienen una herramienta especial para la creación del terreno. Mediante esta se pueden generar grandes extensiones con una calidad asombrosa. El único problema es que se limitan a la temática de la naturaleza. Si se quiere realizar un escenario con una temática diferente hay que recurrir a programas de terceros con los que hacer los modelados necesarios, como unos edificios o unos barcos. Cabe destacar que los propios motores pueden realizar modelados simples. Sin embargo, para crear personas, animales, edificios o infraestructuras y seres vivos en general, es necesario usar programas especializados en el modelado en 3D.

Algunos de los mejores programas para modelar son *Blender*, *Zbrush*, *Maya*, *Lightwave*, *cinema 4D* o *MakeHuman*. Todos ellos tienen un nivel de acabado profesional, por lo tanto, también tienen una curva de aprendizaje considerada.

Por tanto, es importante discernir si es necesario manejar alguno de estos programas, con todo el trabajo que ello implica. Por suerte, existen otras opciones en el mercado. Actualmente no es necesario saber modelar para crear un videojuego, ya que hay muchos diseños en internet. De hecho, los propios motores gráficos suelen ofrecer un mercado para ellos, como es el caso de la *Asset Store* de *Unity*. En esta, tanto usuarios como compañías crean y adquieren *Gameobjects* (clase base para todas las entidades en las escenas de *Unity*), de forma gratuita o pagando. Debido a la gran variedad de opciones disponibles, como personajes, objetos o escenarios, no se considerará necesario el uso de los programas anteriormente nombrados.

En cuanto a las animaciones, estas son el resultado de procesos que consisten en la aplicación de movimientos a imágenes, dibujos o personajes modelados para un videojuego. Se realizan en bucle, donde la secuencia siempre acaba en el mismo lugar, como, por ejemplo, estar de pie, saltar y volver a estar de pie. Las combinaciones posibles a tener en cuenta son inmensas. Siguiendo el ejemplo anterior, el personaje podría correr, sentarse o realizar cualquier otro movimiento programado [25].

En el proyecto a desarrollar el aspecto visual no es primordial, por lo que el uso de animaciones para objetos queda descartado. Por tanto, solo habría que encargarse de animar el avatar, el cual se moverá tal cual lo haga el usuario. Para esta animación se podrían usar las herramientas por defecto de cualquier motor gráfico de los ya mencionados. Por ejemplo, *Unity* cuenta con la herramienta *Animation*. Esta se podría encargar de animar el avatar de forma correcta.

4.4 Dispositivos de captura de movimiento

La captura de movimiento es un sistema que consiste en la medida y almacenamiento de movimientos reales, para inmediata o posteriormente, analizarlos y reproducirlos en un modelo 3D de manera fácil e intuitiva. Para ello, la mayoría de programas que trabajan con animaciones, cuentan con herramientas que permiten la transcripción de información desde algún dispositivo de entrada. Esta información puede ser procesada de forma que el movimiento queda registrado, pudiéndose implantar en cualquier modelado [4].

Se basa en las técnicas de fotogrametría, las cuales consisten básicamente en medir coordenadas en 3D. Su uso se ha extendido principalmente en dos industrias. La industria cinematográfica y la de los videojuegos. En la primera se comenzó a usar en la década de los 90, destacando en películas como *Star Wars Episodio I: La Amenaza Fantasma*. En la segunda, su usó se popularizó por la mejora en los movimientos de los avatares. Actualmente, se está extendiendo su uso a fines deportivos y médicos.

La captura de movimiento ofrece una serie de ventajas frente a la animación por ordenador tradicional. La latencia es mínima, prácticamente es en tiempo real. Los movimientos complejos son mucho más sencillos de realizar y, al igual que las interacciones físicas, también más realistas. Además, la eficiencia es mucho mayor, debido a que se puede animar lo mismo en un tiempo menor.

No obstante, no todos son ventajas. Se requiere un *hardware* y *software* específico para recolectar y procesar la información, lo cual aumenta los costes. El sistema puede requerir condiciones específicas de espacio para que los movimientos a realizar se hagan en el plano de la cámara o cámaras. Los movimientos que no siguen las leyes de la física no se pueden capturar. Si las proporciones del modelado son diferentes a las de una persona se ha de tener especial cuidado con los movimientos, ya que, por ejemplo, unos dedos extremadamente largos podrían atravesar el cuerpo del modelado si la persona que lo representa no lo tienen en cuenta [26].

Existe una gran variedad de sistemas de captura de movimientos, entre los cuales caben destacar los electromagnéticos, los electromecánicos, los ópticos, los inerciales y los de ultrasonidos. Excepto los ópticos, todos los demás requieren de equipamiento para el usuario, ya sean sensores que envíen señales eléctricas, trajes con potenciadores, acelerómetros y giroscopios o emisores de ultrasonidos. Sin embargo, los sistemas ópticos no requieren que ningún elemento esté en contacto con el paciente. Su funcionamiento consiste en la recogida de datos mediante sensores de imagen. Hasta hace poco, era necesario el uso de indicadores en posiciones específicas del cuerpo del usuario, pero con el tiempo, se ha hecho posible la omisión de estos. El rendimiento de este método suele aumentar con la cantidad de cámaras que rastreen al sujeto, ya que se crea una proyección virtual más precisa [27].

La mejora de este método ha sido alcanzada gracias al desarrollo de las cámaras de profundidad, puesto que son estas las que permiten la omisión de los indicadores.

Por tanto, teniendo en cuenta que se va a tratar con pacientes que han sufrido un ictus, y que, en adición, no tienen la necesidad de ponerse sensores por todo el cuerpo, la opción más adecuada para nuestra situación es el uso de un sistema óptico. En los siguientes apartados se analizarán las distintas opciones existentes en el mercado de este sistema.

4.4.1 Kinect

Este dispositivo de captura de movimiento fue diseñado por *Microsoft* para la videoconsola *Xbox 360*. Hoy en día es compatible tanto con los nuevos modelos de consolas de la marca, como con *Windows 7, 8 y 10*. Está basado en la tecnología infrarroja y permite el reconocimiento de gestos y del habla. Por tanto, es capaz de controlar aplicaciones con el cuerpo y la voz. Esto es conocido como la interfaz natural de usuario (NUI). Sin embargo, la característica más interesante para nuestro proyecto es que puede realizar un seguimiento del cuerpo humano en 3 dimensiones [27].

Fue anunciado por primera vez el 1 de junio de 2009, en el E3, bajo el nombre de *Poject Natal*. Sin embargo, no fue presentada hasta marzo de 2010, fecha en la cual se haría oficial su actual nombre, *Kinect*. En junio de 2016, *Microsoft* anunció oficialmente la salida al mercado de su SDK para uso no comercial. Esto sería así, hasta finales de octubre, cuando finalmente fue divulgado que habría una versión comercial de *Kinect for Windows program.*

Cabe remarcar que, en 2014, junto a la salida de la nueva consola de Microsoft, hubo una revisión del sensor, dando lugar a la versión 2.0, la cual mejora en muchos aspectos a su predecesora. Esta nueva versión destaca por su nueva cámara principal (una cámara *time-of-flight*) de alta resolución, la cual permite capturar más detalles con mayor resolución y precisión. Además, la fidelidad en la profundidad detectada es hasta tres veces más efectiva.

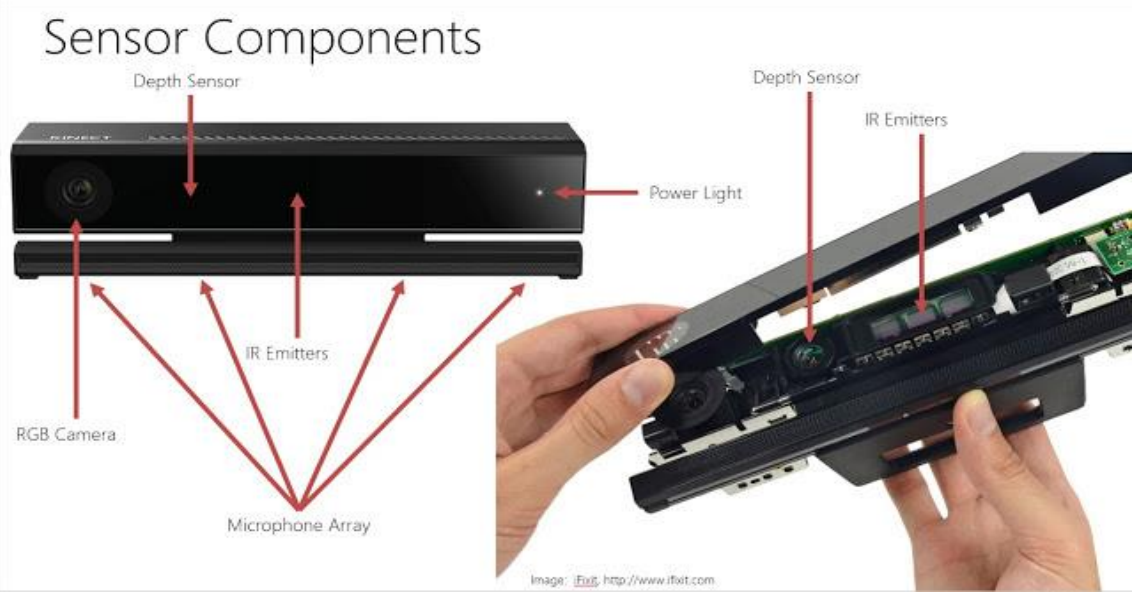

Figura 4.11: Sensores de Kinect.

En cuanto a los componentes que forman el dispositivo, señalados en la **figura 4.11**, tenemos los siguientes:

- Una cámara RGB, con una resolución de 1280x960p y con capacidad de grabar en color.
- Un emisor de infrarrojos (*IR emitter*) y un sensor de profundidad. El primero se encarga de emitir un haz de luz infrarroja, mientras que el segundo es capaz de detectar la que es reflectada. Estos haces reflectados se convierten en información que refleja la distancia entre el sensor y un objeto.
- Un conjunto de 4 micrófonos capaces de grabar sonido. Además, permiten detectar el origen del sonido y la dirección en la que se propagan.
- Un acelerómetro 2G/4G/8G. Este da la opción de determinar la posición actual del Kinect.

También caben resaltar las principales especificaciones del sensor. Tiene un campo de visión de 43º en vertical y 57º en horizontal. Cuenta con un rango de movimiento de 27º verticalmente. Su rango de actuación óptimo es entre 80cm y 4 metros. Funciona a 30 fps (*frames per second*), lo que equivale a que puede procesar hasta 30 fotogramas por segundo. Su acelerómetro está configurado en un rango de 2G, con 1º de error máximo en la posición [28].

Las mejoras de la versión 2.0 son remarcables, por lo que si se eligiese este sensor se usaría esta versión. Destaca un campo de visión un 60% mayor, pasando de 57º en horizontal a 70º y de 43º en vertical a 60º, la posibilidad de detectar hasta 6 personas, frente a las 2 de la primera versión, la adición de 5 articulaciones más de las que se puede saber su posición y orientación, aumentándose así su número a 25, una resolución facial 20 veces mayor, un aumento en la resolución de la cámara a color a 1920x 1080p y de la de profundidad a 512x424p, soporte *multi-app*, que permite a varias aplicaciones usar el mismo sensor a la vez y soporte para *Unity pro* [29].

Su precio ronda los 100€, aunque este puede aumentar si se va a usar en un ordenador debido a la necesidad de adaptadores.

A pesar de esto, la plataforma del pc es apoyada por la compañía, Esto se puede ver en que es posible desarrollar aplicaciones con la librería gratuita que ofrece *Microsoft*, además de con *OpenCV, OpenNI o IIISU SDK*. Además, se pueden usar varios lenguajes de programación: *C++, C# y Visual Basic*. Por todas estas razones, es una opción a tener muy en cuenta para nuestro proyecto [27].

4.4.2 DepthSense 525

Este sensor ha sido diseñado por la empresa belga *SoftKinetic*, que se encarga de desarrollar hardware y software especializado en reconocimiento de gestos y visión 3D. Estos están destinados al entretenimiento digital, la electrónica, la salud, el fitness, la automoción y los S*erious Game*.

Figura 4.12: *DephtSense 525*.

Softkinetic fue fundada en julio de 2007 por matemáticos, especialistas en imagen 3D e ingenieros de software. En febrero de 2012 anunciaron la primera cámara del mundo capaz de rastrear manos y dedos a una distancia de tan solo 15cm. Posteriormente, el 8 de octubre de 2015 la empresa fue adquirida por Sony.

Actualmente poseen la patente de CAPD (*Current Assisted Photonic Demodulator*), una variación del método *Time-of-Flight* (TOF), que permite un rendimiento mejor, mayor cantidad de pixeles y mejor integración que cualquier otra tecnología de TOF en el mercado.

Esta patente está implementada es su modelo más actual, El *DepthSense 525*, que según la propia página web ha vendido más de 15000 unidades en los últimos años. Este cuenta con el módulo DS541A para detectar la profundidad, el más pequeño del mundo con funcionamiento TOF.

Este modelo es capaz de mantener una tasa de 60fps siempre que el objetivo se encuentre en un rango de entre 15cm y 1 metro del sensor. Además, es capaz de funcionar a 15fps a una distancia de 2.5m y a 6fps a una de 4 metros. La resolución de la cámara de color es de 1180x720p, mientras que la cámara de profundidad puede proveer una resolución de 320x240p. Es compatible con *Windows 7 y 8* y se alimenta y transmite la información mediante una sola conexión de USB 2.0.

Anteriormente, permitía trabajar con un middleware denominado *iisu SDK* para el desarrollo de aplicaciones. Sin embargo, este se ha quedado obsoleto. Ahora están disponibles para su descarga en la página web *DephtSense SDK* y *CILib*. El primero

permite el control de la cámara mediante el uso de comandos, mientras que el segundo se encarga del control de gestos simples [30].

Su precio es de 99$, rebajándose este en 10$ por unidad si se adquieren más de 10 dispositivos. Por tanto, es un sensor a tener muy en cuenta por su bajo precio, su tamaño y sus características, aunque habría que analizar si estas son suficientes para nuestro proyecto, ya que, por ejemplo, no cuenta con reconocimiento de cara o de voz.

4.4.3 Xtion Pro Live

Es el primer dispositivo de captura de movimiento exclusivo para ordenador. Está orientado a crear juegos y aplicaciones interactivas con el movimiento. Nace como resultado de la unión alcanzada entre *PrimeSense* y *Asus* para introducirse en el mercado de la captura de movimiento.

Figura 4.13: *Xtion Pro*.

La primera versión de este dispositivo solo contaba con cámara de profundidad. Posteriormente se añadió una cámara a color y actualmente, su hardware permite la detección de gestos del cuerpo entero, o de ciertas partes en específico, como pueden ser las manos. También tiene la capacidad de reconocer sonidos, por lo que es posible el uso de la voz para controlar ciertos parámetros. Cabe destacar que esta última versión es compatible con Unity3D.

En cuanto a sus especificaciones, hay que tener en cuenta que su alimentación se realiza por un puerto USB 2.0. La resolución de la cámara de color es de 640x480p, mientras que la de profundidad provee 320x240. La primera de estas alcanza los 30fps, mientras que la segunda llega a los 60fps. Tiene un campo de visión de 58º en horizontal y 45º en vertical. Su rango de actuación es entre 0.8 y 3.5 metros. Su precio ronda en torno a los 150€.

Es compatible con las versiones anteriores a *Windows 7* (ésta incluida), *Linux* y *Android*, siendo esta último solo bajo petición. Además, tiene un kit para el desarrollo de software: *OPEN NI SDK bundled*.

Por último, destacar que permite varios lenguajes de programación dependiendo de la plataforma en la que se use. En *Windows* permite C++ y C#, en *Linux* C++ y C y en *Android java* [31].

4.4.4 Creative Senz3D

Dispositivo con cámara de profundidad y reconocimiento de gestos para PC. Es capaz de detectar gestos con la mano y movimientos de la cabeza, por lo que permite interactuar de manera única con el ordenador. También puede detectar el contorno de la cara, detectando así al usuario que lo usa.

Figura 4.14: *Creative Senz3D.*

Permite el uso del software *FastAccess 3D facial recognition*, el cual, usando la tecnología de reconocimiento de cara y el sensor de 3 dimensiones, permite acceder a ciertas aplicaciones sin el uso de una contraseña. Esto quiere decir, que son las métricas de la cara de una persona las que funcionan como contraseña. Por lo que, usando el sensor, solo haría falta mostrar la cara para acceder a la aplicación requerida.

En cuanto a sus prestaciones, es capaz de capturar imágenes a 720p en color, mientras que puede captar la profundidad con una resolución de 320x240p. En ambos casos, puede alcanzar hasta los 30fps. Tiene un FOV (*Field of View*) de 74º y un rango de funcionamiento de entre 15cm y 1 metro. En cuanto a su conexión, esta se realiza mediante un USB 2.0. También incorpora un par de micrófonos con reducción de ruido y cancelación de eco. Finalmente, destacar que cuenta con *Nuance Dragon Assistant 1.5*, que permite el uso de comandos de voz para interactuar [32].

Como conclusión, mencionar que es posible desarrollar aplicaciones y juegos gracias a librerías como *Intel Perceptual Computing SDK*, *IISU SDK de Softkinetic*, *OpenCV* y *PCL*. Además, es compatible con Windows 7 y 8. Todo esto, sumado a que su precio es de 200$, lo hace una opción muy interesante.

4.4.5 Vicon

Por último, es preciso mencionar este sistema por ser considerado como uno de los mejores sistemas de captura de movimiento en la actualidad. Además, es el usado en el hospital donde se realizarán pruebas con los pacientes. Por tanto, es interesante analizarlo para tener una mejor perspectiva a la hora de compararlo con un sistema de menor costo.

Se basa es un conjunto de cámaras que analiza un espacio de captura, en el cual se situará el objeto o la persona a analizar, tal y como se puede ver en la **figura 4.15**. Es necesario que existan una serie de indicadores que reflejen los haces de luz. Cada cámara de *Vicon* es una cámara digital estándar, rodeada de un anillo de *LEDs*. Estos son los que envían los haces de luz infrarroja, que rebotan en los marcadores y son detectados por las cámaras de *Vicon*. Los marcadores están hechos de un material especial, que maximiza la reflexión haciendo posible que no haya confusiones con otros materiales reflectantes como el cristal de un reloj.

Cada una de las cámaras devuelve una imagen en 2 dimensiones cada fracción de segundo a *MX Ultranet HD*. La conexión de este dispositivo con las cámaras se muestra en la **figura 4.15**. Basándose en la geometría de la habitación y la posición de las cámaras se pueden obtener las posiciones de todos los marcadores. Es necesario que un mínimo de dos cámaras detecte un marcador para poder posicionarlo. Una vez hecho esto, las coordenadas de todos los puntos son enviadas al ordenador, el cual, mediante el software de Vicon Nexus, permite la visualización de los resultados y el uso de estos [33].

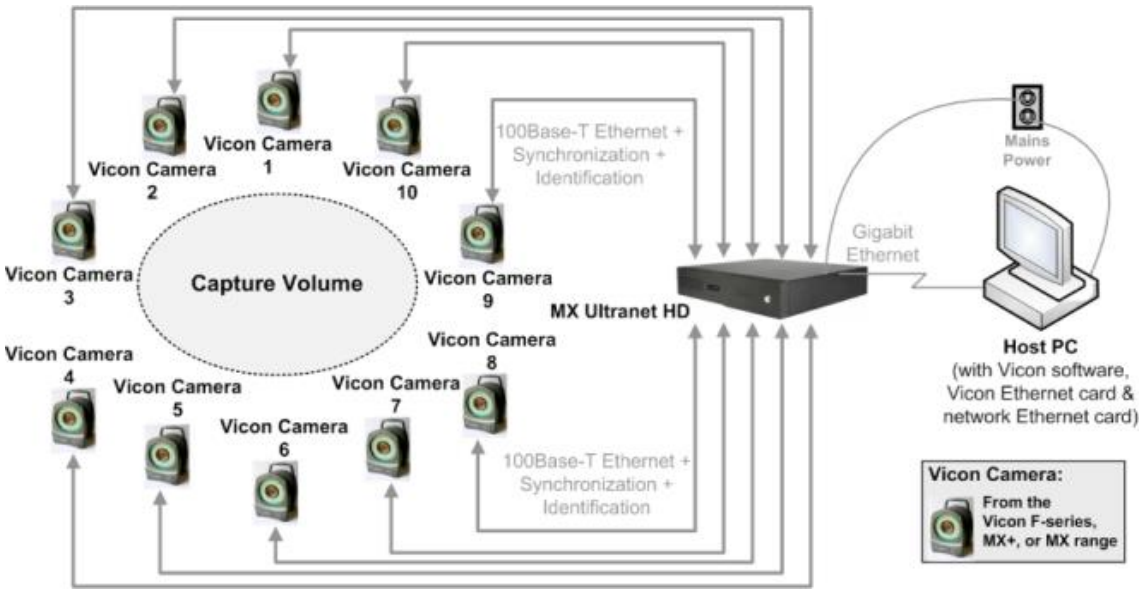

Figura 4.15: Típica organización del sistema *Vicon*.

Este sistema puede ser utilizado en una gran variedad de industrias. En la propia página web del sistema (*https://www.vicon.com/*) se señalan la biomecánica y el deporte, las ciencias clínicas, el entretenimiento (destacando el cine en este sector), la realidad virtual o el seguimiento de objetos.

Existen varios modelos de cámaras en el mercado, con características específicas. Yendo desde el modelo V16, con una resolución máxima de 4096x4096 pixeles (16 Megapíxeles), pudiendo llegar a procesar hasta 2000 imágenes por segundo, hasta el modelo v1.3 que cuenta con tan solo 1.3 Megapíxeles y un procesamiento de hasta 250 imágenes por segundo. No se ha encontrado el precio exacto del dispositivo, pero se han encontrado reseñas en las que el precio variaba entre unos miles y más de cien mil euros, dependiendo de la instalación.

4.5 Comparativa y elección de tecnologías

Una vez analizadas todas las opciones en el mercado, es el momento de compararlas y elegir cual es la que más se adapta a nuestras necesidades.

4.5.1 Motores gráficos

En primer lugar, cabe destacar que no se dispone de unos conocimientos lo suficientemente amplios como para crear un juego de cero. Por tanto, las opciones relacionadas con programar desde cero quedan descartadas. Esto deja como candidatos a los motores gráficos y a las herramientas de desarrollo previamente nombradas.

Partiendo del hecho de que el principal objetivo del proyecto es obtener las métricas del paciente, es importante que el método usado sea flexible. Esto quiere decir, que se permita la experimentación para lograr el procedimiento óptimo a usar. Según este requisito, podemos descartar *GameMaker: Studio*, ya que no posee la variabilidad suficiente.

También se puede descartar *Adobe AIR*, puesto que para obtener los resultados deseados habría que tener una base de programación mucho más amplia de la que se dispone. Esto se podría solventar con el tiempo suficiente, pero dado que la cantidad de tiempo también es limitada, no es una opción viable.

Tras este análisis inicial, solo han quedado como opciones válidas los motores gráficos, ya que permiten unos resultados realmente buenos, sin necesidad de tener una gran base de conocimientos. Para elegir entre uno de ellos hay que tener en cuenta diferentes aspectos [17]:

- Tipo de juego que se quiere realizar. Los distintos motores se especializan en distintos ámbitos, por lo que algunos tienen mejor rendimiento en 2D o en 3D o incluso no habilitan uno de estos modos.
- Tipo de licencia. Si se quiere modificar el código será necesario un motor *OpenSource*. Cabe destacar que los comerciales dan más facilidades.
- Plataforma a la que se dirige el juego. Hay que comprobar que el motor es compatible con la plataforma requerida.
- Herramientas a utilizar. Hay que comprobar que el motor es compatible con *software* de terceros y que cuente con las herramientas internas necesarias como para crear nuestro contenido.
- El lenguaje de programación. Es importante que use uno conocido, o si no es el caso, fácil de aprender.
- La documentación y el soporte de la comunidad. Es imprescindible para aprender a usar el motor, responder las dudas que surjan y tener ejemplos en los que basarse.

Teniendo todo esto en cuenta, se podría descartar el motor *CryEngine V*, ya que no tiene soporte para Mac. Esto es importante, dado que solo se va a desarrollar el juego

para ordenador, es importante que todos los usuarios puedan acceder a él, independientemente del sistema operativo que usen.

Tras esto, solo quedan dos opciones entre las que elegir, *Unity* y *Unreal Engine*. Actualmente son los dos motores más importantes de la industria, tal y como refleja la revista *Develop*, que sitúa a *Unreal* como el mejor motor y a *Unity* como el segundo [18]. O la página web *Statista*, que preguntó en 2014 a cientos de desarrolladores en el Reino Unido por el motor gráfico que estaban usando, dando como resultado que *Unity* era el más usado, seguido por *Unreal Engine* (**figura 4.16**) [34].

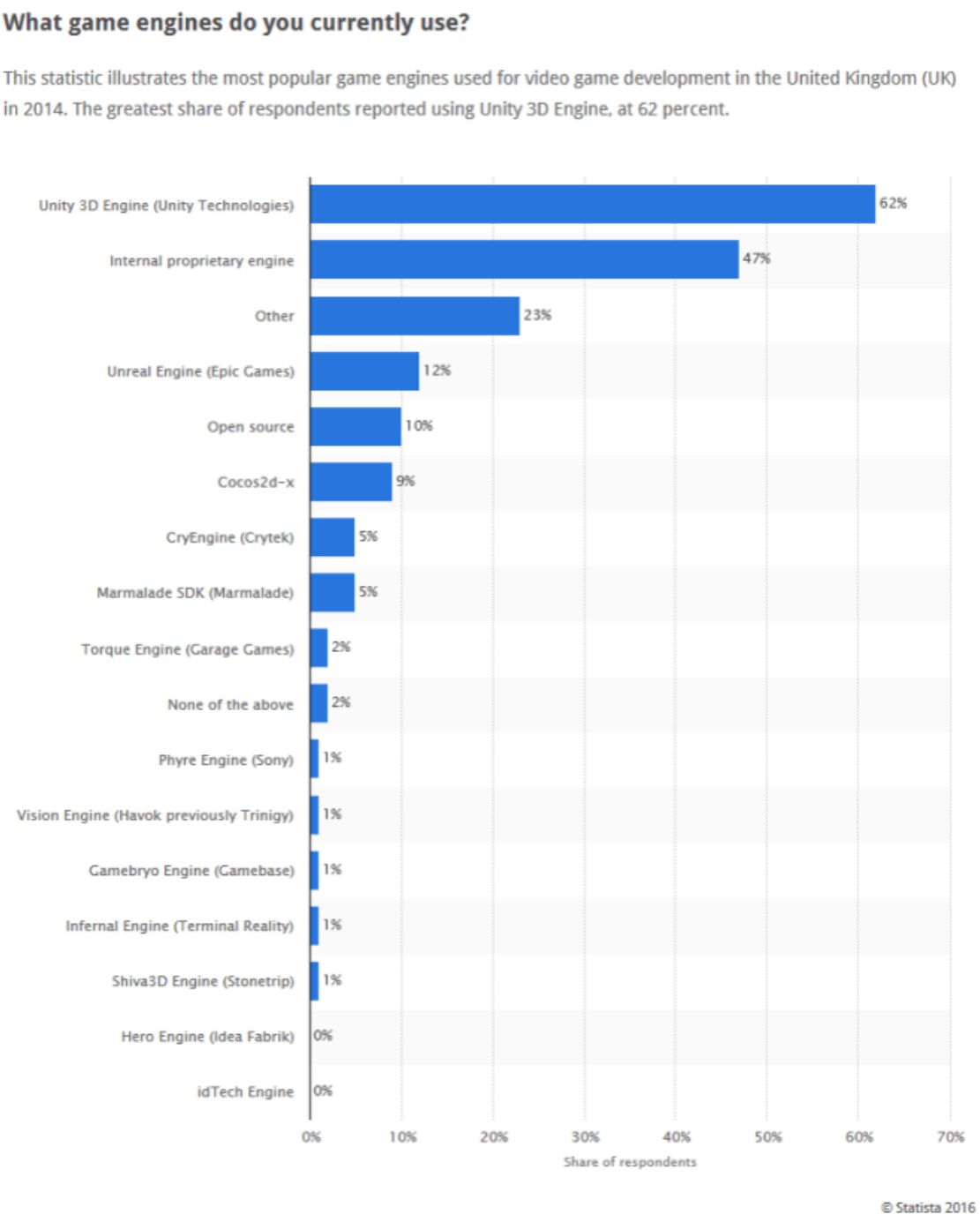

Figura 4.16: Motores gráficos más usados en el Reino Unido en 2014

Con todo esto podemos concluir que estamos ante las dos mejores opciones posibles. Sin embargo, es necesario elegir una, la más adecuada.

Un aspecto importante a destacar es que *Unity* permite el uso de C# y *UnityScript*, mientras que *Unreal* trabaja con C++. En mi caso, ya había trabajado con anterioridad con C++, por lo que ya es un lenguaje conocido, ahorrando así parte del periodo de aprendizaje. Sin embargo, C# es un lenguaje muy similar, por lo que el aprenderlo no sería un gran inconveniente.

En cuanto a la adquisición de modelados en 3D. En *Unity* existe la *Asset Store*, mientras que en *Unreal* existe *Marketplace*. Ambas herramientas son oficiales y permiten la descarga de contenido que puede ser usado en el juego. En este punto, *Unity* tiene la ventaja, puesto que ofrece una cantidad mayor de contenido y, sobre todo, permite filtrar este por precio. Esto es un gran problema en *Unreal*, ya que para encontrar el contenido gratuito es preciso indagar en foros donde la propia comunidad indica cómo conseguirlo.

Cabe remarcar la comunidad y la documentación en ambos motores. A fin de que un principiante pueda aprender el funcionamiento de todas las herramientas dadas es preciso una gran cantidad de información. Esta es dada en la documentación. Ambos poseen una extensa y detallada explicación de cada una de las funciones que ofrecen. Además, disfrutan de las dos comunidades más grandes, lo cual abastece de información sus correspondientes foros, en los cuales se puede resolver prácticamente cualquier duda.

Llegados a este punto, se puede apreciar que estamos ante dos motores con unas especificaciones que se adaptan de forma precisa a nuestras necesidades. Sin embargo, hay dos factores más a tener en cuenta. La compatibilidad con el sensor que se escoja, es decir, que se puedan usar conjuntamente y que la implementación sea sencilla; y la elección realizada por compañeros de años anteriores, ya que sus proyectos pueden servir de guía para este.

Por tanto, la elección del motor gráfico se hará conforme a los resultados en la comparación de los sensores. Ya que no se puede elegir uno sin tener en cuenta el otro.

4.5.2 Dispositivos de captura de movimiento

Una vez llegados a este punto, es necesario comparar todos los sensores mencionados anteriormente. Para facilitar esta comparación se muestran las principales características en la **tabla 4.1**.

En primer lugar, hay que tener en cuenta el rango de acción de los dispositivos. En nuestro proyecto, los usuarios podrían estar situados desde una distancia de 1 metro hasta una distancia lo suficientemente lejana como para que el sensor detectase el cuerpo entero. Por este motivo el sensor C*reative Senz3d* queda descartado. Su rango no supera el metro por lo que no podría detectar bien al usuario.

En cuanto al *DephtSense 525*, este también tiene un rango limitado a un metro si se quiere que sus fotogramas por segundo no disminuyan de 60. Sin embargo, puede detectar a una distancia mayor con una reducción significativa de estos, como ya se explicó anteriormente. En nuestro caso, esto no nos conviene, ya que queremos obtener

las métricas de los pacientes en cada instante de tiempo, con la mayor fluidez posible. Además, requiere de luz ambiente para su correcto funcionamiento, mientras que otros sensores como el *Kinect* no. Por lo tanto, este también queda descartado.

	KINECT 2	DephtSense	Xtion	SENZ3D™
Resolución a color	1920x1080	1180x720	640x480	1280x720
Resolución en profundidad	512x424	320x240	320x240	320x240
Rango	Entre 80cm y 4m	Entre 15cm y 1m	Entre 80cm y 3.5m	Entre 15cm y 1m
FPS	30	60	30 en RGB 60 en profundidad	Hasta 30
Consumo energético	Alimentación adicional al USB	<2.5W	< 2.5W	<2.5W
Software	OpenNI, OpenCV, IISU SDK, Kinect for Windows SDK	DepthSense SDK y CILib	OPEN NI SDK bundled	Intel Perceptual Computing SDK, IISU SDK, OpenCV y PCL
Compatibilidad	Windows	Windows	Windows Linux Ubuntu Android (bajo petición)	Windows
Precio	≈100€	99$	≈ 150€	≈200$

Tabla 4.1: Comparativa de sensores de captura de movimiento

Por consiguiente, la elección se dará entre *Kinect* y *Xtion*. Si nos fijamos en las especificaciones de ambos dispositivos podemos observar como *Kinect* sobresale en prácticamente todos los aspectos, tendiendo una mejor cámara, tanto de color como de profundidad, una mayor cantidad de software con la que trabajar y un precio inferior. No obstante, también requiere de un adaptador específico, que le provea corriente y permita la conexión con un ordenador mediante USB 3.0. Además, solo es compatible con *Windows*, por lo que se perdería al público que trabaje con *Mac* o *Linux*. Por tanto, ambos tienen sus ventajas y sus inconvenientes.

Sin embargo, es necesario resaltar que *Kinect* es un dispositivo mucho más afianzado en el mercado, lo que se traduce en un mejor soporte y una documentación mucho más extensa. Y, puesto que no se dispone de conocimientos sobre el tema, es algo a tener muy en cuenta. En adición a todo esto, los proyectos realizados en años anteriores ya se decantaron por el uso de *Kinect*. Esto implica, que se podrían resolver dudas dentro

del propio equipo de trabajo y que el dispositivo ya se encuentra en el laboratorio, lo que reduciría los costes de adquisición de otro sensor.

De esta forma, por todas las ventajas mencionadas, el dispositivo de captura de movimiento elegido es *Kinect 2.0*.

4.5.3 Modelo final

Si analizamos la compatibilidad entre el *Kinect 2.0* y las dos opciones de motor gráfico restantes, nos encontramos con que ambos tienen una implementación prácticamente idéntica y de forma casi inmediata. Por tanto, el factor decisivo en la elección del motor gráfico es el motor usado en años anteriores. Y este es el *Unity*.

Ambos servirían para realizar el proyecto, pero debido a que otros proyectos anteriores, de los que se puede aprender, usaron este motor, me he decantado por su uso. Una vez se ha elegido el motor y el sensor a utilizar, es necesario aprender de ambos.

En cuanto al motor, la mejor forma de aprender a usarlo es siguiendo los tutoriales que la propia empresa pone a disposición del usuario en su página web. Con estos, se aprenden los conceptos básicos del motor y las herramientas que ofrece. Una vez se completan esos tutoriales se puede empezar a realizar pequeños juegos en los que se aplique lo aprendido. Posteriormente, ya se puede empezar a trabajar en el proyecto. A pesar de todo lo aprendido, es necesario documentarse e ir viendo tutoriales de forma continuada, conforme este avanza, puesto que la cantidad de contenido que ofrece el motor es tan grande, que irán surgiendo dudas a lo largo del desarrollo.

Si nos referimos al sensor, este tiene un problema ya mencionado anteriormente. Y es que, al estar diseñado en primer lugar para consola, necesita de un adaptador especial para conectarlo a un ordenador. Este problema fue solventado en 2014, cuando *Microsoft* lanzó *Kinect adapter for Windows*, permitiendo así el uso de este dispositivo en la susodicha plataforma.

Además, *Microsoft* también ofreció un *plugin* para permitir a los desarrolladores usar una gran variedad de herramientas con las que crear aplicaciones que hagan uso de su *hadware*. Este plugin es *Kinect for Windows SDK 2.0*. Este nos permite usar el dispositivo con *Windows*, pero no solo eso, también incorpora demostraciones, ejemplos y programas básicos para facilitar la comprensión del funcionamiento del sensor.

Los más importantes son:

- *Kinect Configuration Verifier*. Permite comprobar que el ordenador que se está usando tiene los requerimientos mínimos para que Kinect funcione correctamente. Esto, además, nos permite saber si la conexión se está realizando correctamente.
- *Kinect Studio*. Proporciona una interfaz en la que se puede monitorear, grabar y obtener datos del sensor.

- *Body Basics*. Demuestra cómo usar la herramienta que permite obtener y visualizar los fotogramas en los que se rastrea el cuerpo del usuario, es decir, permite observar el esqueleto que detecta el *Kinect*.
- *Color Basics*. Demuestra cómo usar la herramienta que permite obtener y visualizar fotogramas a color. También enseña como guardar un fotograma como un archivo .png.
- *Coordinate Mapping basics*. Demuestra como eliminar el fondo, dejando solo la persona. Es el mismo efecto que el de un croma verde.
- *Depht Basics*. Demuestra cómo usar la herramienta que permite obtener y visualizar fotogramas de profundidad.
- *Face Basics*. Enseña cómo obtener puntos, orientaciones y expresiones de la cara de una persona.
- *Infrared Basics*. Demuestra cómo obtener y visualizar fotogramas en infrarrojo.
- *Speech Basics*. Demuestra cómo usar el reconocimiento de voz, permitiendo reconocer frases simples en inglés.

Hay que remarcar como punto adicional que la mayoría de las opciones vienen en varios lenguajes de programación, de forma que cada persona puede elegir el más adecuado para su caso.

5 Diseño desarrollado

En este apartado se desarrollará la solución propuesta para el problema planteado. Para ello, se llevará un análisis en profundidad del *Serious Game* creado. En primer lugar, se abarcarán los aspectos más generales de este, como su finalidad o las mecánicas de juego utilizadas. A continuación, se explicarán de forma detallada los elementos más importantes del juego, como la implementación del sensor Kinect o la adquisición y el guardado de datos. Por último, se analizarán las distintas escenas que componen el juego y la utilidad y uso de cada una.

5.1 Aspectos generales

El objetivo de este juego serio es el de mejorar la calidad de vida de una persona con problemas de movilidad, que han surgido como consecuencia de haber sufrido un ictus. Para ello, se realizará en primer lugar una evaluación del estado del paciente. En esta se realizarán varios movimientos de la escala *Fugl-Meyer*.

A continuación, el paciente podrá practicar su movilidad en un minijuego que consiste en esquivar tablas que se acercan al avatar. Estas tablas tienen formas especialmente pensadas para que solo se puedan esquivar realizando posiciones específicas, es decir, incitan al paciente a trabajar movimientos en los que tiene problemas, con el objetivo de incrementar su movilidad.

La evaluación se puede realizar tantas veces como se quiera, al igual que el minijuego de práctica. En ambos casos, se consigue una puntuación al final. Esto se ha realizado así para que el jugador sienta la necesidad de superarse y, además, pueda ver su progreso reflejado en sus resultados. De esta forma, se pretende acentuar el interés y la motivación del paciente.

Otro requerimiento que se tiene que considerar viene dado por el rango de edades de los jugadores a los que se enfoca el videojuego. Debido a que esta dolencia es más común en gente de edad más avanzada, el juego ha de ser sencillo y fácilmente entendible. Lo cual significa que la interfaz debe de ser clara, al igual que la forma en la que se juega. Esto también es importante para el fisioterapeuta, ya que un manejo sencillo siempre es beneficioso. Para ello, se han simplificado los menús y se han implementado video y audio para cada movimiento a realizar.

Además, los parámetros con los que se juega en cada ocasión deben ser modificables por el terapeuta, ya que debe haber un progreso en la dificultad si se quieren obtener buenos resultados. Estos pueden ser variados en el menú principal, al inicio o durante una sesión. Entre ellos destacan el tiempo máximo para realizar un movimiento, o la velocidad a la que se acercarán las tablas a esquivar.

En cuanto a la información que se recopila, esta se podría diferenciar en dos tipos. Por un lado, se genera un Excel por cada paciente y juego (*Fugl-Meyer* o Esquivar las

tablas). En ellos se reúne toda la información relacionada a la sesión. En el caso de la evaluación, se guardan los resultados obtenidos en cada ejercicio, así como el tiempo que se ha empleado. En el minijuego, se registran los parámetros con los que se ha trabajado, la puntuación y los errores totales. Por el otro lado, se crea un archivo de Excel por cada articulación, en el que se muestra la posición que ha tenido esta por cada instante de tiempo. También se guardan los ángulos que se han obtenido entre ciertas articulaciones. Estos datos han servido para discernir la precisión con la que se detectan los distintos movimientos, y, por tanto, su validez.

Para concluir, resaltar que las características que han guiado el desarrollo:

- Interfaz sencilla y mecánicas de juego fáciles de entender.
- Capacidad de motivar al paciente mediante un sistema de puntuaciones.
- Progreso en la dificultad mediante la variación de parámetros.
- Capacidad de rastrear el cuerpo y los movimientos del paciente.
- Evaluación del estado del paciente.
- Adquisición y guardado de datos para su posterior análisis.

5.2 Consideraciones del diseño

El desarrollo de un videojuego es un proceso indudablemente complejo, que requiere del afinamiento de una gran cantidad de detalles. Debido a ello, no se explicarán todos los pasos seguidos con precisión. Esto quiere decir que, por ejemplo, el uso del Kinect o la adquisición de datos serán totalmente aclarados, mientras que el movimiento de las nubes o la colocación de los elementos gráficos de fondo, como rocas o árboles, serán ignorados por su carencia de importancia. Su única función es la de hacer el ambiente más ameno, solo son elementos decorativos.

5.2.1 Conexión con Kinect

Tal y como se explicó en el apartado de tecnologías, *Unity* y *Kinect* son plenamente compatibles. Sin embargo, es necesario saber cómo se realiza la implementación del sensor para poder trabajar con él en el motor gráfico.

En primer lugar, es necesario descargar e instalar *Kinect for Windows SDK 2.0*, puesto que este software contiene los drivers oficiales proporcionados por *Microsoft*. Además, también proporciona códigos de ejemplo y varias APIs *(Application Programming Interface).* Con esto, ya es posible el uso de Kinect en el ordenador. Esta herramienta facilita enormemente la creación de aplicaciones para los desarrolladores, entre distintos factores porque cuenta con soporte oficial.

A continuación, si se accede a la *Asset Store* de *Unity* se puede encontrar un paquete llamado *Kinect with MS-SDK* y otro que se llama *Kinect v2 Examples with MS-SDK*. En ellos se encontrarán todos los elementos necesarios para desarrollar un videojuego usando el *Kinect*. Hay que destacar las escenas de ejemplo, en las que se ejemplifican los diferentes usos de los *script*s que se proporcionan. Para la realización del S*erious Game*, hay que importar el segundo de estos paquetes en el proyecto, ya que serán necesarios varios de los *script*s incluidos en este.

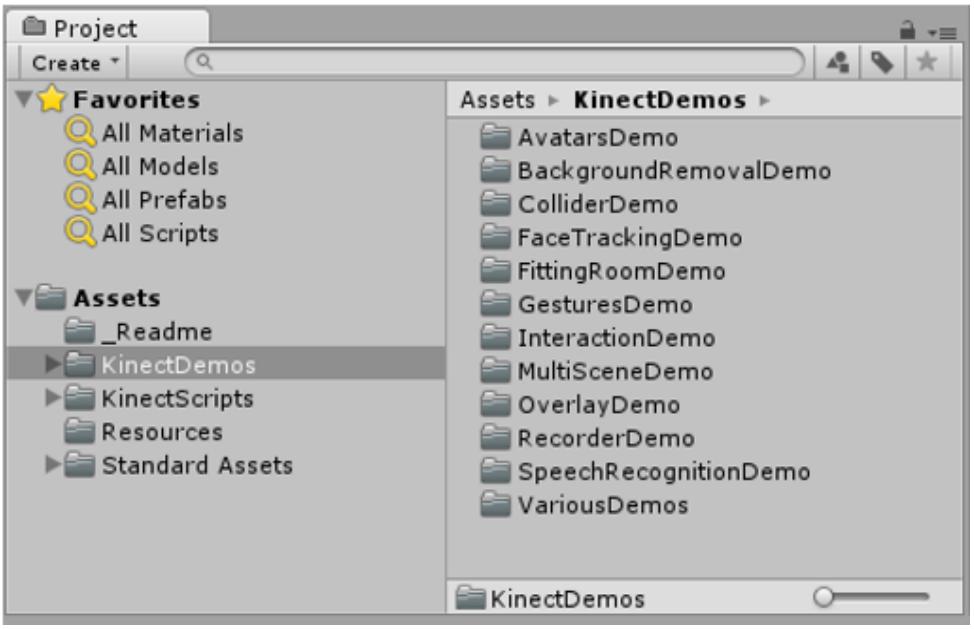

Figura 5.1: Contenido del paquete *Kinect v2 Examples with MS-SDK*

Concretamente serán necesarios *AvatarController*, *AvatarControllerClassic*, *KinectManager*, *GetJointPosition*, *JointPositionView* y *JointOrientationView*. Los dos últimos solo son usados durante el periodo de desarrollo del juego para ver la calidad de las medidas tomadas, por lo que no son realmente imprescindibles.

El más importante de todos ellos es el *KinectManager.cs*. Este *script* es el que contiene el código que establece y gestiona la relación entre el sensor y el motor gráfico. En la **figura 5.2** se pueden observar los parámetros que se pueden personalizar, como el número de jugadores a detectar o las opciones del tipo de imagen a mostrar (profundidad, color, mostrar esqueleto rastreado…). La mejor opción es implementarlo en un *GameObject* vacío, debido a que es un controlador global y, por tanto, es recomendable no adjuntarlo a ningún otro objeto. Cabe remarcar que este *script* ha sido modificado para que el juego no empezase hasta que se haya detectado a un jugador, de esta forma se puede evitar que antes de que el paciente esté listo se comience con las pruebas.

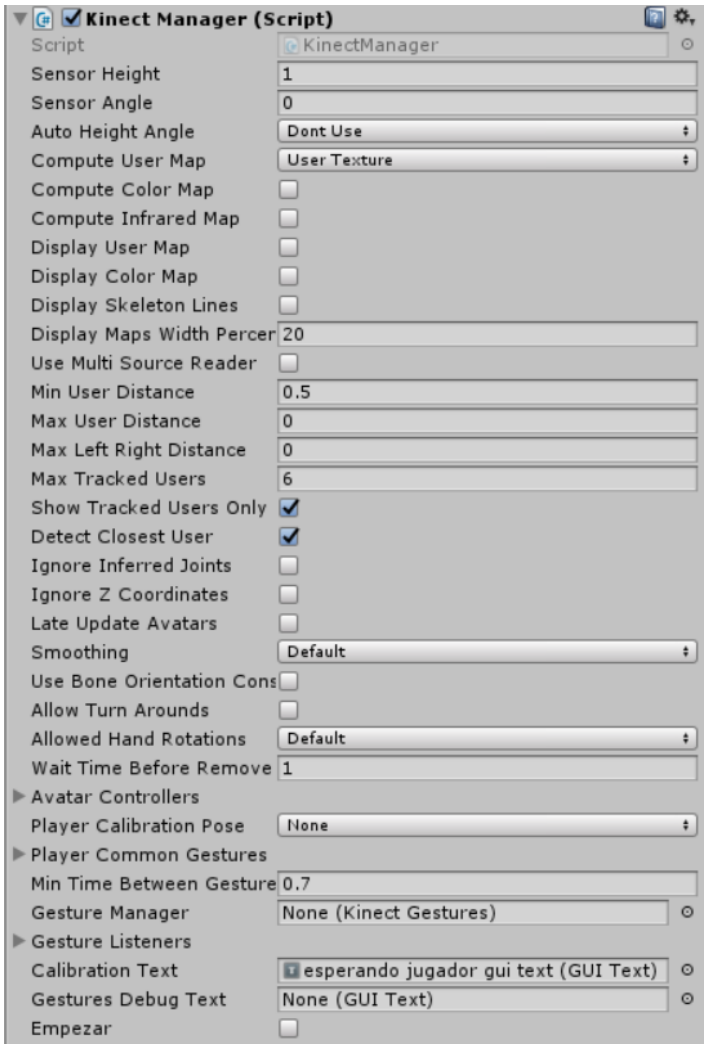

Figura 5.2: Parámetros graduables del *KinectManager*.

En cuanto al *Avatar Controller* y *el AvatarControllerClassic*, ambos sirven para transmitir el movimiento del jugador al avatar del juego. El primero de ellos lo hace con todo el cuerpo, es decir, la posición del avatar será exactamente la misma que la que esté detectando el sensor. Sin embargo, existe un problema con este método. Si el usuario se sienta, o alguna articulación sale del campo de visión del *Kinect*, este deja de detectar correctamente el estado del usuario, provocando que el avatar haga movimientos irreales. En las pruebas realizadas, el error más común que se ha obtenido ha sido en las piernas. Cuando el jugador se sienta, muchas de las articulaciones que detecta el sensor se alinean o se juntan, lo que da lugar a que haya que eliminar esas medidas.

Justamente para eso se ha creado el segundo de estos. El 'controlador clásico' permite introducir las articulaciones que se quieren rastrear, manteniendo una posición predefinida para las demás. El sensor sigue detectando todo el cuerpo, pero solo el movimiento de la parte seleccionada se transmite al avatar. Este *script* ha sido utilizado en la escena en la que se evalúa al paciente con el *Fugl-Meyer*, ya que muchos de los ítems se realizan en una posición sentada. En la **figura 5.3** se pueden apreciar los parámetros ajustables de ambos, pudiéndose apreciar, además, las articulaciones que se han seleccionado para ser detectadas. Es importante destacar que la posición neutra establecida es de pie, con los brazos en una posición relajada, pero ligeramente separados

del cuerpo. Por tanto, cualquier articulación que no se quiera rastrear se mantendrá en la posición que tenga durante esta postura neutral.

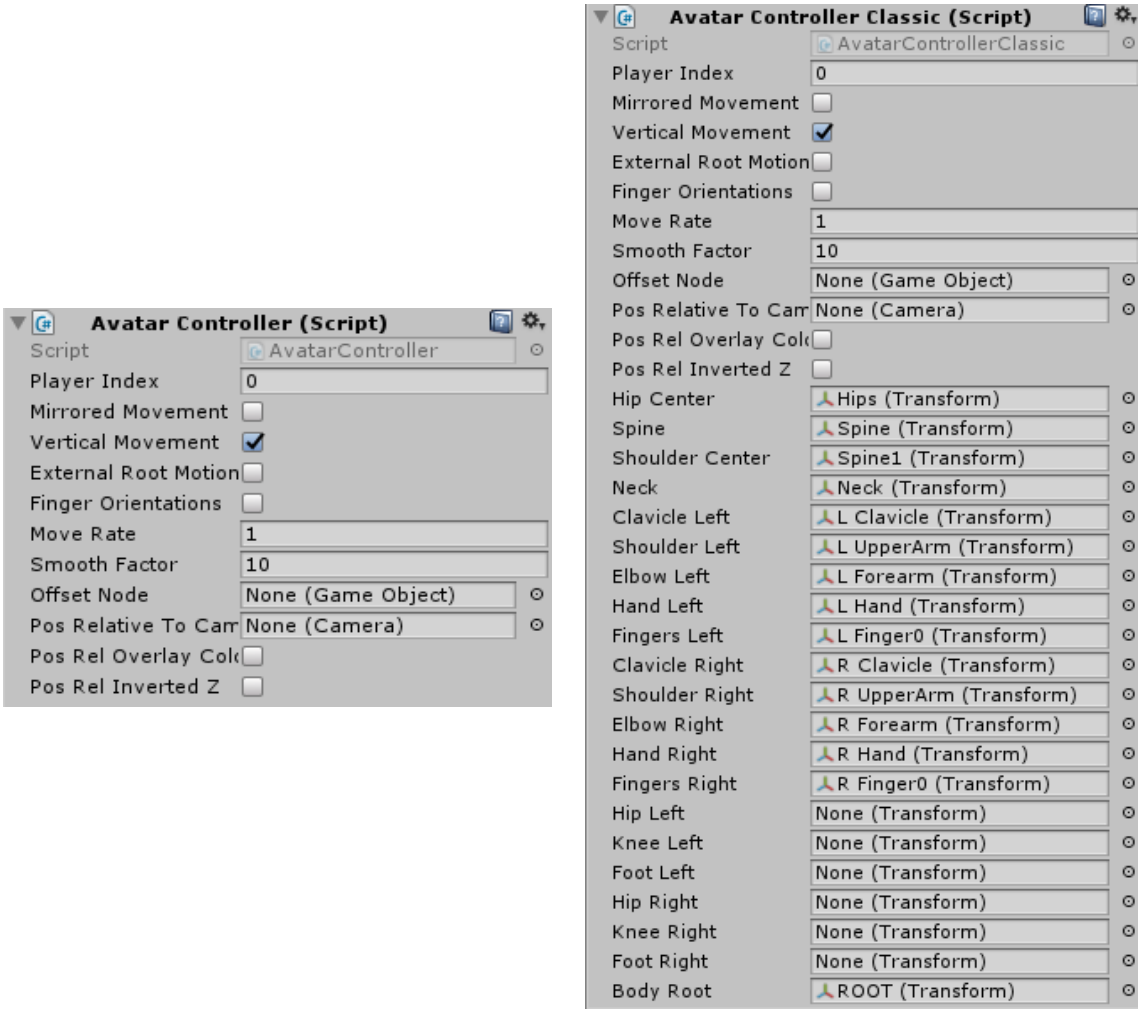

Figura 5.3: Parámetros regulables en *AvatarController.cs* y *AvatarControllerClassic.cs*

5.2.2 Avatar y entorno

Una vez se ha llegado a este punto, ya se tienen todos los elementos que se encargarán de agregar movimiento al avatar. Sin embargo, aún falta el avatar. Hay distintas opciones para conseguir uno, se pueden crear mediante programas de modelado como *Maya* o *Blender*, se pueden importar desde otras plataformas o se pueden descargar de la *Asset Store*. La última opción es la más sencilla y la que garantiza una calidad mayor en nuestro caso.

En la *Store* que proporciona *Unity* se pueden encontrar una gran cantidad de modelos. Para este proyecto se han seleccionado dos de ellos, que se pueden encontrar como: *Modern Female Professional Secretary* y *Sci Fi Officer Captain*. Ambos modelados son muestras gratuitas del paquete *All Star Characters.* Además, también se

usará un tercer modelado, que será el creado el año anterior por Jaime Herreros [35] mediante *Make Human*. El jugador podrá elegir entre estos tres personajes para jugar.

Figura 5.4: Modelados de los tres avatares

Tras importar un modelado en la escena en la que se esté trabajando, se puede especificar el tipo de *rig*, es decir, la definición de avatar, en nuestro caso esta debe ser humanoide. Al hacerlo, *Mecanism* (herramienta de animación propia de Unity*)* se encarga de cuadrar la estructura ósea del avatar con una estructura humana. En muchas ocasiones, este paso se realiza automáticamente. Este proceso genera un *sub-asset*, en el cual se podrán modificar las características del avatar.

El avatar, por lo tanto, debe de tener dos componentes, el *AvatarController.cs* del que ya se ha hablado y el *Animator*. Este último es el encargado de controlar el sistema de animación de *Mecanism*. Debe de llevar asignado en el parámetro "avatar" el *sub-asset* correspondiente, ya que sino los movimientos podrían no ser los adecuados.

Cabe resaltar antes de continuar con el siguiente paso, que los avatares están formados por "*joints*" o articulaciones. Estas son *GameObjects* vacíos localizados en puntos estratégicos del cuerpo, de forma que cada uno de ellos se correspondan con su articulación real. Esto implica que, al ser *GameObjects*, tienen una posición y una orientación específica en cada instante de tiempo. El *Kinect* se aprovecha de este hecho y transmite la posición y orientación de las articulaciones reales del usuario a las virtuales, consiguiéndose así que ambos movimientos sean casi idénticos. No obstante, el *Kinect* no puede detectar tantas articulaciones como tiene el avatar, de forma que habrá partes del cuerpo que no se muevan, un ejemplo de esto podrían ser los dedos. Mientras que el avatar tiene una articulación por falange, el *Kinect* solo puede detectar si la mano está abierta, cerrada o en una posición intermedia. Concretamente el sensor puede detectar 25 articulaciones, entretanto, el avatar de la chica tiene 52. La mayoría de estas se pueden ver en la **figura 5.5**.

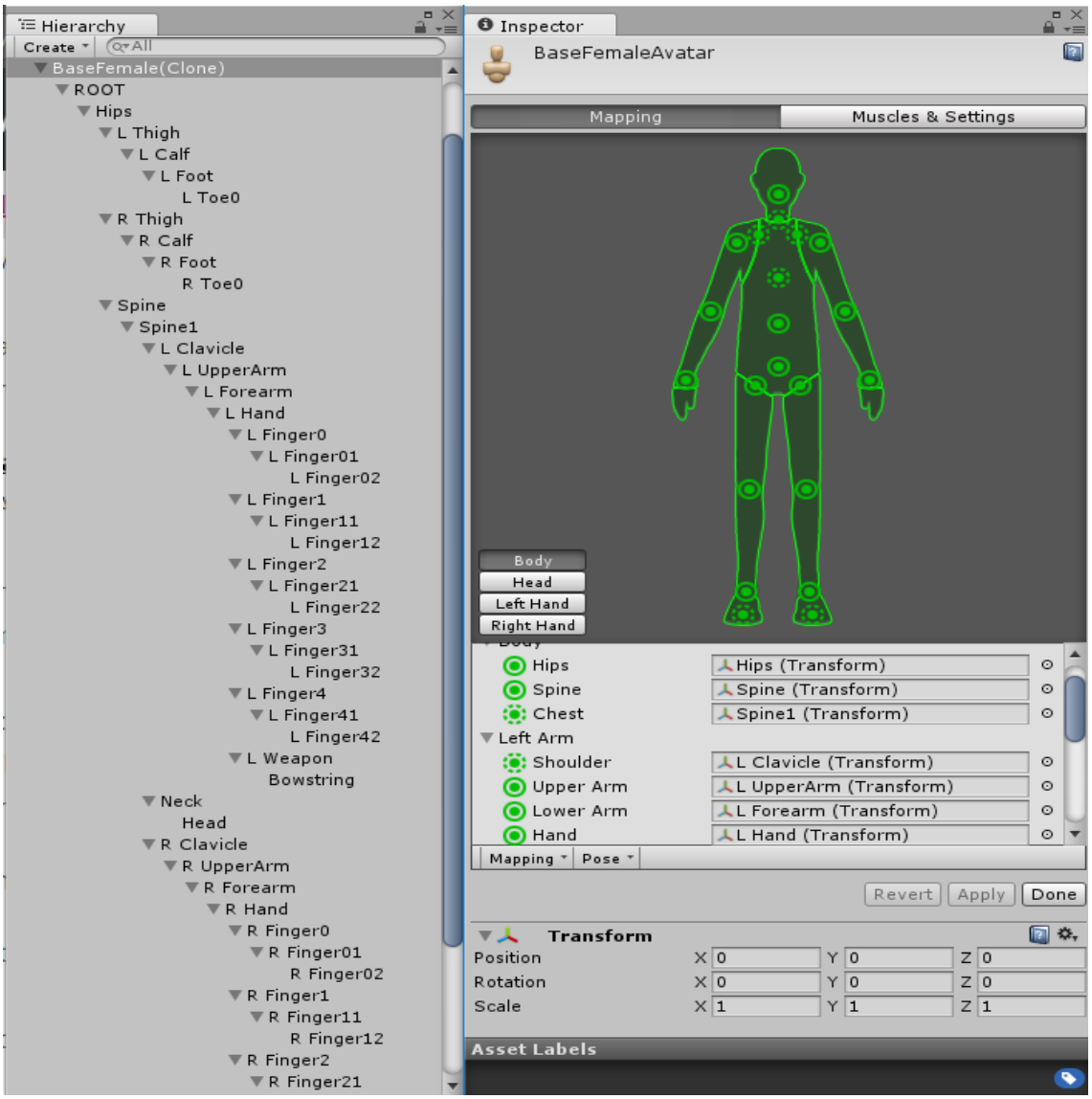

Figura 5.5: Configuración del *sub-asset*

Con esto, el avatar ya cumple todos los requisitos como para moverse emulando los movimientos del jugador.

El siguiente paso para la creación del *Serious Game* es la creación de un entorno con el que el jugador pueda interactuar. Este estará formado por una gran variedad de *GameObjects*, algunos interactivos y otros simplemente decorativos. También contará con *Canvas*, que son los elementos que Unity utiliza para mostrar las interfaces de usuario, por tanto, se usarán para mostrar las puntuaciones, los movimientos a realizar…

Existen diferentes formas de crear este entorno. Se pueden diseñar diferentes *GameObjects* con las funciones que proporciona Unity, también se pueden importar objetos simples y complejos e incluso diseños que por sí mismos se pueden considerar un entorno gráfico completo, como podría ser un bosque, una ciudad o el interior de un gran edificio. El método más sencillo para importar estos elementos es mediante la *Asset Store*, aunque también existe la posibilidad de importarlos desde otras aplicaciones. Por último, al igual que en el modelado del Avatar, se pueden desarrollar modelos personalizados en programas como *Blender* o *Maya*.

En nuestro caso, todos los modelados del entorno se han descargado de la *Asset Store*. Una vez importados en el proyecto, hay que incorporarlos a la ventana de *Hierarchy*, en la cual se muestran todos los elementos de la escena. Una vez en la escena, se puede modificar tanto el tamaño, como la posición o las propiedades del objeto.

Figura 5.6: Escena del minijuego de esquivar

Figura 5.7: Escena de la evaluación *Fugl-Meyer*

5.2.3 Obtención y guardado de información

Parte del objetivo del proyecto es la adquisición de datos del paciente para su posterior análisis. De esta forma, se puede medir más rigurosamente la evolución del paciente. Si se analiza el paquete anteriormente mencionado, *Kinect v2 Examples with MS-SDK,* se pueden encontrar varios *script*s que nos servirán para llevar a cabo este cometido.

En primer lugar, tenemos los *scripts JointPositionView.cs* y *JointOrientationView.cs*. Estos sirven para visualizar la posición y la orientación de una articulación a lo largo del tiempo. En los parámetros variables mostrados en la **figura 5.8** se puede observar como la articulación a rastrear es una de las opciones. Por tanto, simplemente copiando este *script* numerosas veces y poniendo cada vez una articulación distinta, se puede mostrar por pantalla la posición y orientación de todas las articulaciones que puede detectar el *Kinect*.

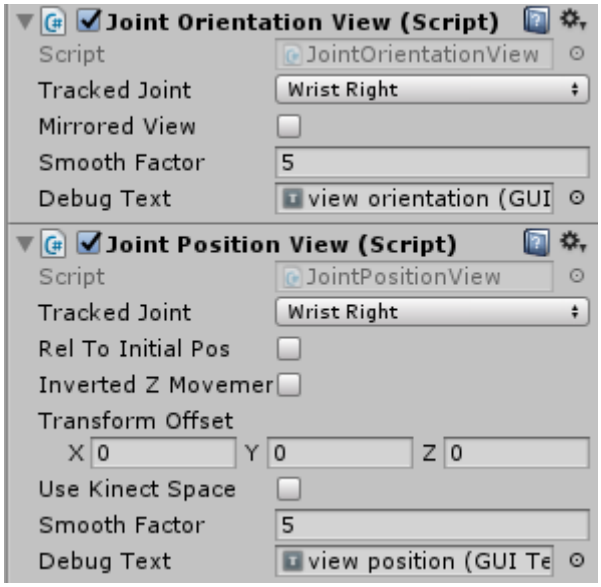

Figura 5.8: Parámetros variables de *JointPositionView.cs* y *JointOrientationView.cs*

Sin embargo, solamente con esto no se está obteniendo ningún tipo de dato. Objetivamente, estos *scripts* únicamente han sido utilizados para visualizar los datos directamente mientras se juega, consiguiéndose así un consecuente ahorro de tiempo.

Para la adquisición de datos se usará el *script GetJointPosition.cs*. Este permite el guardado de las coordenadas cartesianas de las diferentes articulaciones por instante de tiempo. De forma que, una vez finalizado el juego, se puede acceder a unos archivos que contengan la posición en la que ha estado cada articulación en cada momento. Estos datos pueden ser realmente útiles, ya que con ellos se puede conocer el punto más alto alcanzado por el paciente, así como al más bajo o el más lejano, la velocidad a la que mueve cada articulación, puesto que se conoce el espacio que recorre y el tiempo que tarda, o incluso los ángulos que realiza en cada movimiento.

Para los ángulos se puede crear un *script* propio, de forma que no sea necesario un tratamiento posterior de los datos para conocer los ángulos. El que se ha desarrollado

funciona obteniendo las coordenadas de dos articulaciones y realizando una serie de operaciones con ellas.

Estas operaciones se basan en que un punto en 3 dimensiones se puede situar en el espacio por medio de tres ángulos, los cuales son los que forma al proyectarse sobre cada uno de los planos xy, xz e yz. En la **figura 5.9** se puede observar la proyección de dos puntos sobre el plano xy.

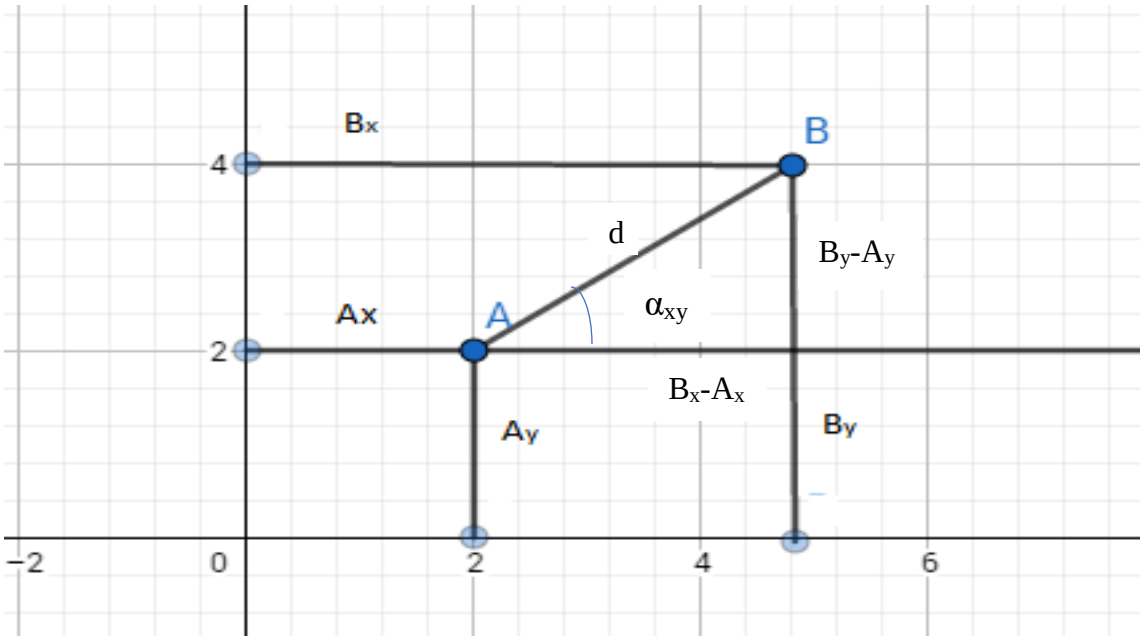

Figura 5.9: Puntos A y B proyectados sobre el plano xy

Si se restan las distancias en el eje *x* y las distancias en el eje *y* de ambos puntos, obtenemos la distancia que separa a los puntos en esa proyección. Con esta distancia podemos usar la siguiente fórmula:

$$\tan \alpha_{xy} = \frac{\sin \alpha_{xy}}{\cos \alpha_{xy}} = \frac{(B_y - A_y)/d}{(B_x - A_x)/d}$$

A partir de la cual se puede despejar el ángulo que se busca:

$$\alpha_{xy} = \tan^{-1}\left(\frac{(B_y - A_y)}{(B_x - A_x)}\right)$$

Si se hace las mismas operaciones para cada una de las proyecciones se obtienen los ángulos que forman esos dos puntos con cada uno de los planos cartesianos. En *Unity* esto se puede realizar con la función `Mathf.Atan2()` y la función `Mathf.Rad2Deg`. La primera de estas sirve para calcular el arcotangente en radianes, mientras que la segunda transforma los radianes a grados.

Con esto ya se tendría la posición y los ángulos que se requieran por instante de tiempo. Sin embargo, es necesario explicar el método mediante el cual se pueden guardar estos datos. Tras investigar por diversos foros técnicos se ha concluido con que la mejor opción es guardar los datos en formato CSV (*comma-separated values*). Este formato es

compatible con *Microsoft Office*, está destinado a la representación sencilla de tablas y consiste en la separación de líneas por saltos de línea y de columnas por comas. Por lo tanto, se adapta perfectamente a las necesidades del proyecto.

En cuanto a la forma de guardado de los datos, está se podría diferenciar en dos tipos. Por un lado, se guardarán los datos relacionados con la posición y los ángulos de las articulaciones. Estos datos se guardarán de forma que se creará un Excel con el nombre de la articulación y el número de intentos que se han realizado. Un ejemplo podría ser *ElbowRight_intento2.csv*. Por otro lado, se creará un Excel por cada paciente y tipo de juego, es decir, que cada paciente tendrá dos Excel de este tipo. En ellos se mostrarán los parámetros con los que se ha trabajado, la fecha y hora de la sesión, así como su número y todos los datos del paciente. Hay varias opciones para que Excel reconozca bien el formato, la más sencilla es siguiendo estos pasos: abrir una hoja de Excel nueva, ir a la pestaña de datos y seleccionar desde el texto/CSV, elegir el archivo a visualizar y seleccionar cargar. En las **figuras 5.10 y 5.11** se muestra cómo se verían los datos guardados tanto en el minijuego como en la evaluación mediante el Fugl-Meyer.

	Column1	Column2	Column3
2	Nombre del paciente	Lado afectado	Comentarios
3	juan	derecho	se cansa rapido
4			
5			
6			
7	Fecha y hora de la sesion	06-09-2017 17:16:28	
8	Sesion	1	
9	Tiempo mÃ¡ximo permitido por movimiento	3	
10			
11	Movimiento	PuntuaciÃ³n obtenida	Tiempo empleado
12	Elevacion frontal del hombro izquierdo	1	3.10
13	Elevacion frontal del hombro derecho	1	3.11
14	Elevacion lateral del hombro izquierdo	2	0.51
15	Elevacion lateral del hombro derecho	2	0.51
16	Flexion del codo izquierdo	0	3.09
17	Flexion del codo derecho	0	3.05
18	Extension del codo izquierdo	0	3.06
19	Extension del codo derecho	0	3.07
20	Rotacion interna del hombro izquierdo	0	3.04
21	Rotacion interna del hombro derecho	0	3.05
22	Rotacion externa del hombro izquierdo	0	3.08
23	Rotacion externa del hombro derecho	0	3.09
24			
25	PuntuaciÃ³n Total obtenida	6	
26	Tiempo empleado	40.88	
27			
28			
29	Fecha y hora de la sesion	06-09-2017 17:18:14	
30	Sesion	2	
31	Tiempo mÃ¡ximo permitido por movimiento	3	
32			

Figura 5.10: Excel creado para la evaluación *Fugl-Meyer*

	Column1	Column2	Column3
2	Nombre del paciente	Lado afectado	Comentarios
3	Jorge	izquierdo	buena estamina
4			
5			
6			
7	Fecha y hora de la sesion	09-09-2017 18:38:17	
8	Sesion	1	
9	Cantidad de tablas por ronda	1	
10	Tiempo de aparicion entre tablas	3	
11	Numero de rondas	1	
12	Tiempo de espera entre rondas	4	
13	Puntuacion total	0	
14	Fallos totales	1	
15			
16			
17			
18	Fecha y hora de la sesion	09-09-2017 18:40:50	
19	Sesion	2	
20	Cantidad de tablas por ronda	1	
21	Tiempo de aparicion entre tablas	3	
22	Numero de rondas	1	
23	Tiempo de espera entre rondas	4	
24	Puntuacion total	10	
25	Fallos totales	0	

Figura 5.11: Excel creado para el minijuego de esquivar

Como se puede ver en las imágenes toda la información de un paciente sobre un juego irá en el mismo Excel indicando mediante la fecha y número de sesión cuando se ha realizado cada ejercicio. Los nombres para estos archivos serán el del paciente más una extensión que puede ser "_Fugl_Meyer" o "_EsquivarGame". Los archivos mostrados, por ejemplo, se llaman "juan_Fugl_Meyer" y "Jorge_EsquivarGame", por lo que la localización de los datos para el terapeuta es lo suficientemente simple.

En cuanto a los archivos que se crean con la información relativa a la posición y al ángulo de las articulaciones, estos simplemente están formados por una columna, en la que se muestra el instante de tiempo, y otras tres con las coordenadas x, y, z de cada articulación en el caso de las posiciones y con los ángulos de las diferentes proyecciones en el caso de los ángulos.

En el anexo se encuentra el código necesario para realizar este tipo de guardado a archivos CSV, así como el *script* usado para calcular los ángulos.

5.3 Módulos del *Serious Game*

El juego desarrollado se puede dividir en diferentes escenas que, en conjunto, dan lugar al resultado final. Cada una de estas escenas tiene un propósito en concreto y se han desarrollado teniendo en cuenta diferentes factores. Por tanto, es importante analizar cada una de ellas por separado.

5.3.1 Menú principal

La primera escena con la que el usuario se encuentra al iniciar la aplicación es el menú principal. Este está dividido en diferentes pantallas, que van dirigiendo al jugador a través de las diferentes alternativas. En la primera pantalla se encuentran las cuatro opciones principales: Jugar, Opciones, Instrucciones y Salir.

Como su propio nombre indica la última permite salir de la aplicación, mientras que la tercera muestra una serie de instrucciones que servirán tanto al jugador como al terapeuta para entender el funcionamiento del juego.

La segunda, llamada "Opciones", permite modificar los parámetros con los que se trabajará. Además, también permite introducir el número de la sesión y la dirección de memoria en la que se quiere que se guarden los archivos al final de cada partida. No obstante, en caso de que no se ponga ninguna dirección, los datos se guardarán automáticamente en la carpeta que contiene los archivos del juego. En la **figura 5.12** se pueden observar todas las variables que se pueden alterar. Las variables de cada juego se muestran debajo del título de este, y su valor mínimo viene determinado por condiciones de diseño. Por ejemplo, la cantidad de rondas en el juego de esquivar no puede ser menor que una y el tiempo para realizar un movimiento en la evaluación mediante el *Fugl-Meyer* no puede ser menor a tres segundos, ya que no habría tiempo suficiente como para indicar las instrucciones de cada movimiento.

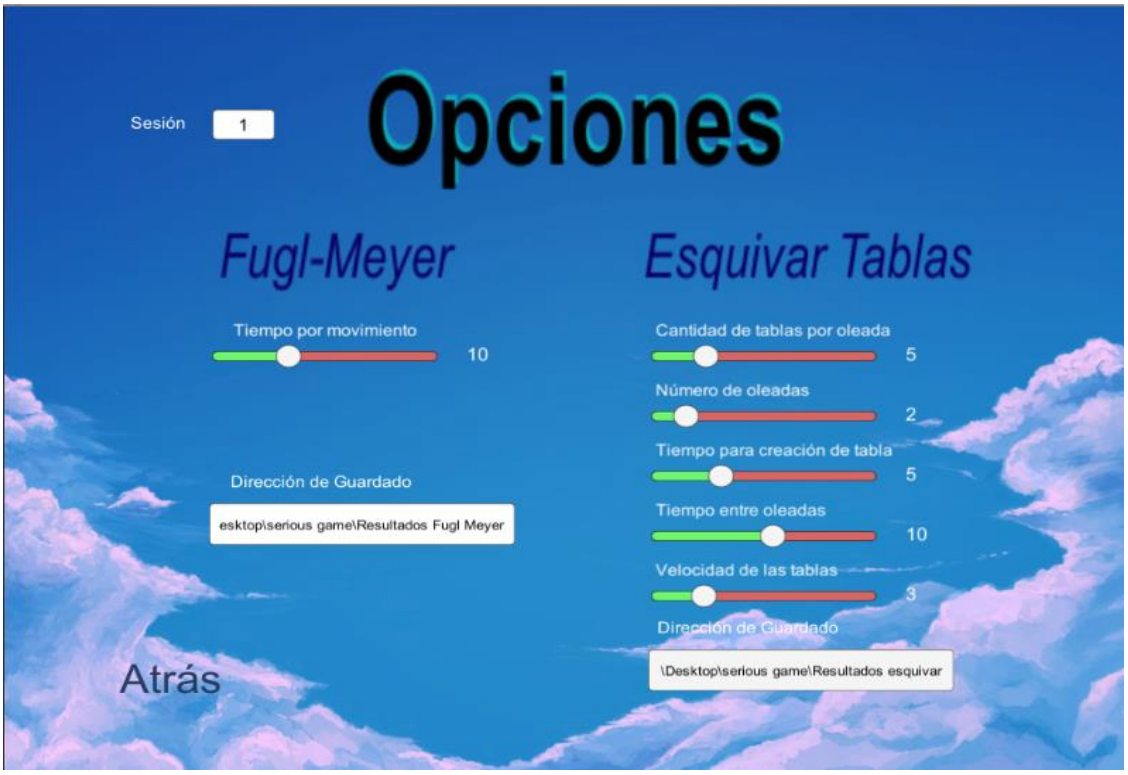

Figura 5.12: Menú de Opciones

Por último, se encuentra el botón de Jugar. Si se pulsa este, aparecerán dos nuevas opciones: Nuevo Jugador y Cargar Jugador. Si se pulsa el primer botón aparecerá la pantalla mostrada en la **figura 5.13**. Esta contiene tres *Input Fields*, en los cuales se

introducen los datos del paciente. El primero será el nombre, y será utilizado como el identificador de usuario para las siguientes sesiones, por lo que no se podrá repetir. Además, también se podrá introducir el lado afectado del usuario y los comentarios que el terapeuta considere necesarios. Finalmente, también se puede elegir el avatar con el que se va a jugar durante la sesión.

Si se da el caso de que ya se ha registrado anteriormente, no será necesario volver a introducir toda la información de nuevo, se puede pulsar el botón de "Cargar Jugador", el cual mostrará una pantalla, similar a la de la **figura 5.14**, en la que el elemento principal es una lista de jugadores dinámica. Esta lista va añadiendo de forma automática a los nuevos jugadores registrados.

Las dos opciones cuentan con un botón de salir para volver al menú principal y otro de confirmar, con el cual los datos del usuario se guardarán y se avanzará a la siguiente pantalla. En esta se muestran las distintas opciones de juego existentes. Estas son la evaluación y el minijuego de esquivar. Una vez se elige el modo de juego se muestran unas pequeñas instrucciones sobre este y se da comienzo el juego. Cabe destacar que estas pantallas también cuentan con el botón de volver al menú principal

Figura 5.13: Menú de Nuevo Jugador

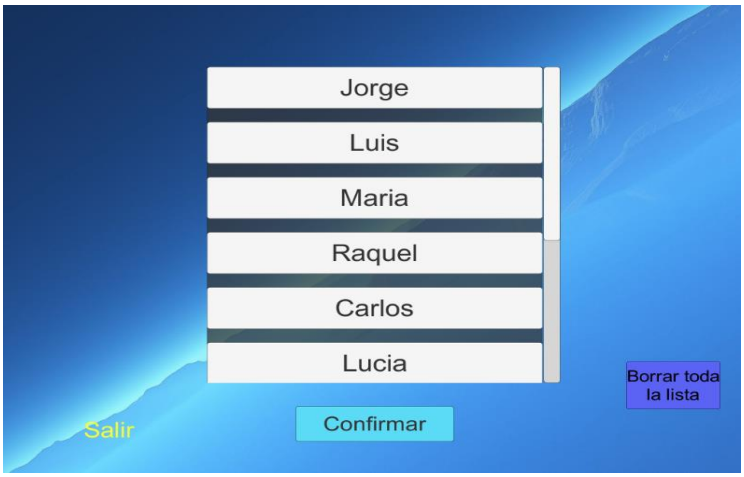

Figura 5.14: Menú de Cargar Jugador

Diagrama de flujo

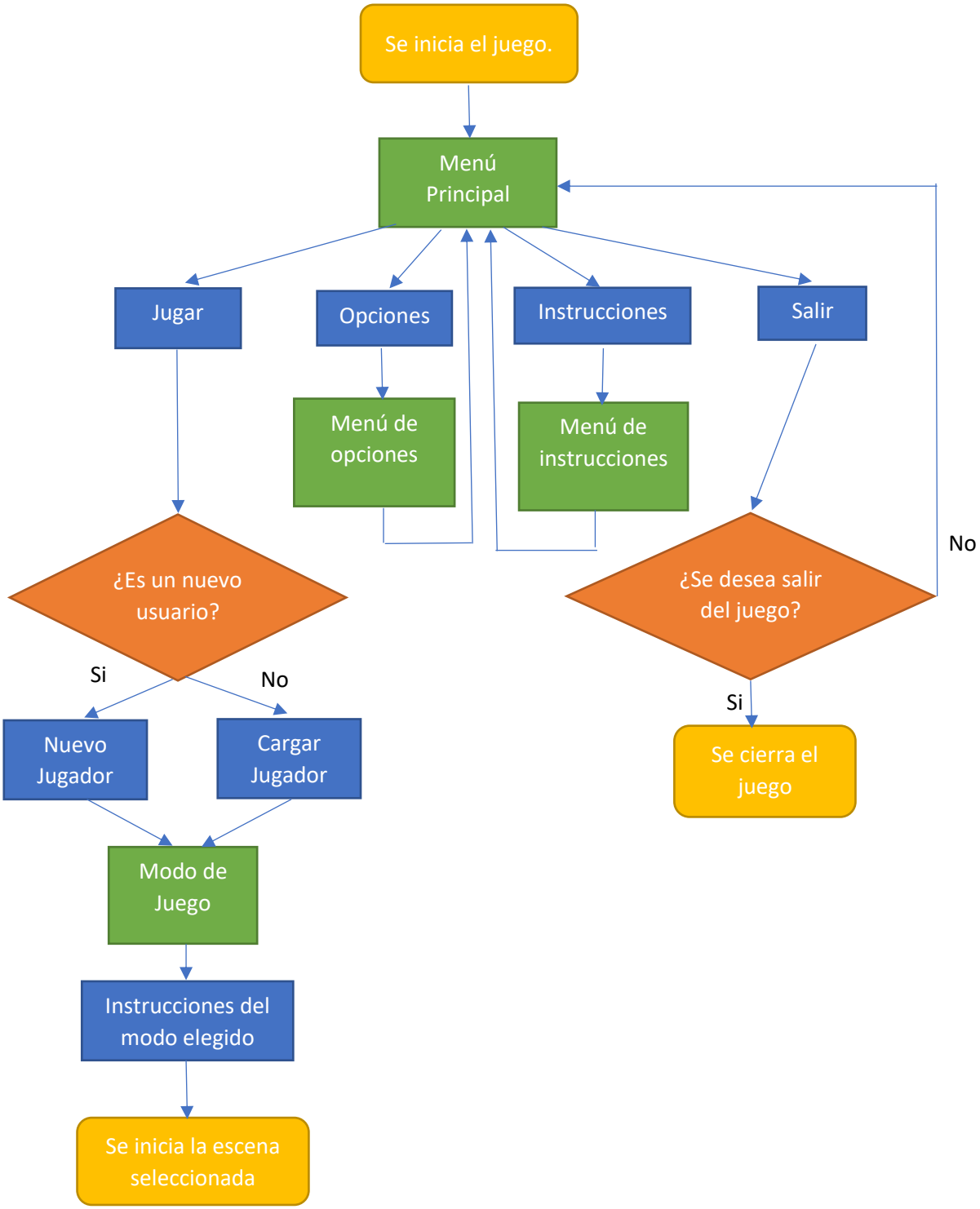

Figura 5.15: Diagrama de flujo del Menú principal

5.3.2 Juego de esquivar

Si se elige la opción de acceder al minijuego de esquivar cargará otra escena en la cual se ha desarrollado este. En primer lugar, cabe resaltar el botón de salir, situado en la zona inferior izquierda de la pantalla, el cual permite volver en cualquier momento al menú principal.

Además de este, hay otro elemento que se muestra según se accede a la escena, el mensaje de "esperando al jugador". Este mensaje se mostrará hasta que el Kinect detecte a un usuario y por lo tanto pueda comenzar el juego. Una vez un usuario se haya detectado desaparecerá el mensaje e iniciará el juego. El primer obstáculo no aparecerá hasta que no hayan pasado dos segundos. Este periodo se ha fijado para dar un margen de preparación antes de comenzar con la prueba.

A partir de este momento los obstáculos empezarán a aparecer tal y como se haya establecido en el menú de opciones. Si no se ha introducido ningún valor, los parámetros usados serán los de la última sesión realizada. Las variables que permiten ser modificadas son:

- La cantidad de obstáculos por oleada
- El número de oleadas
- El tiempo de aparición entre obstáculos
- El tiempo entre oleadas
- La velocidad de los obstáculos

Figura 5.16: Captura del minijuego de Esquivar los obstáculos

Objetivo y mecánicas de juego

En cuanto al funcionamiento del minijuego, este consiste en esquivar todos los elementos que se acerquen al jugador. Esto se puede realizar moviendo el cuerpo del usuario, de forma que el avatar también se mueva y pueda evitar ser golpeado. Los obstáculos son modelados con formas específicas. Han sido creados mediante las propias herramientas de *Unity* y el elegido para ser lanzado al jugador es seleccionado aleatoriamente entre todos los existentes. Esto se hace para que el número de variaciones sea el más alto posible.

La forma de producir y enviar estos objetos viene determinada por las variables previamente mencionadas. De esta forma, los obstáculos se irán enviando en diferentes grupos de una cantidad fija durante la partida. Por tanto, se pueden entrenar una gran variedad de aptitudes. Si se busca un entrenamiento duradero e intenso se puede trabajar con una gran cantidad de oleadas, un pequeño tiempo de aparición entre obstáculos y una gran velocidad de estos. Sin embargo, si se busca el otro extremo, y se quieren sesiones cortas y de baja intensidad, se pueden reducir el número de grupos a uno, minimizar la velocidad y aumentar el tiempo de aparición. Por consiguiente, el entrenamiento posee una progresión y variedad adecuada para un gran rango de usuarios.

Programación

Si se produce una colisión entre el avatar y el obstáculo *Unity* es capaz de detectarlo gracias a los *Colliders*. Estos son componentes invisibles que utiliza el programa para detectar colisiones físicas. Sin ellos los *GameObjects* solo se muestran en el juego, pero no se puede interactuar con ellos físicamente. Cabe resaltar que estos requieren un alto grado de sencillez en su geometría, puesto que son muy exigentes a nivel de consumo de recursos. Para el juego en cuestión, los *Colliders* se han utilizado tal y como se muestra en la **imagen 5.17** con uno de los obstáculos. Están representados mediante prismas huecos formados por aristas de color verde.

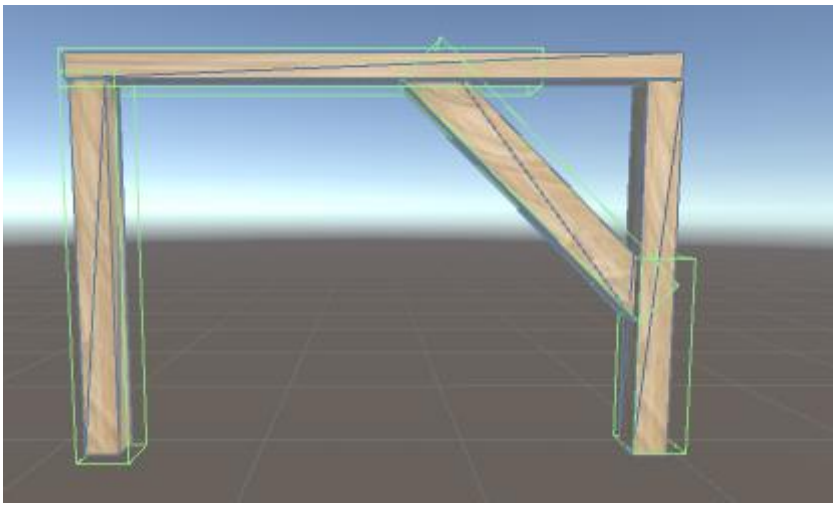

Figura 5.17: Ejemplo de obstáculo con sus *colliders*.

Además, también se han usado en los avatares disponibles, ya que, si no se encuentran en ambos objetos, el choque no se detectará. Un ejemplo de la implementación de *colliders* en los avatares se muestra en la **imagen 5.18**.

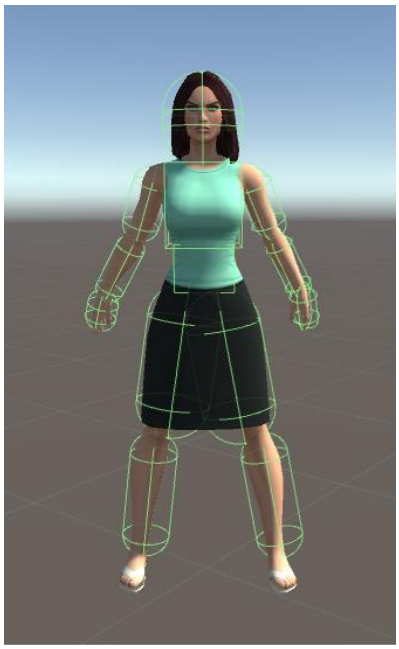

Figura 5.18: Ejemplo de *colliders* en un avatar.

Un aspecto muy importante es que para la función que van a cumplir los *colliders* en esta escena no hay que marcar la opción de *is trigger*. Esto permitirá que la interacción entre el personaje y el obstáculo sea la correcta. La función *is trigger* sirve para que se detecte como un objeto entra dentro del volumen de otro, pero sin provocar una colisión. En nuestro caso se está buscando, no sólo que se sepa que ha entrado en el espacio de otro, sino que el obstáculo reaccione al golpe recibido. Por tanto, no se activará esta función.

También es importante añadir un *rigidbody* a los *GameObject* que posean un *collider*, ya que sino no serán detectados por *Unity*. En el caso de los obstáculos, solo hay que añadir el componente en cuestión, mientras que para los avatares hay que activar la opción *is kinematic*, puesto que sino no seguirán el movimiento del avatar cuando este reaccione al *Kinect*. Vale la pena nombrar que se ha añadido un *script* que permite el movimiento de los obstáculos en dirección al jugador y que, al chocar con él, se activa la gravedad para las tablas, lo que da un aspecto más realista al videojuego.

En cuanto a la puntuación, esta funciona sumando un valor de diez puntos por cada obstáculo que pueda esquivar el jugador. Para ello, se ha colocado otro *collider* detrás del jugador que permite la cuenta de estos. Este si debe tener activada la opción *is trigger*. Si el jugador choca con el obstáculo, este último desaparece a los dos segundos, para no impedir el correcto desplazamiento de los que vienen a continuación, y se cuenta un fallo. Al final de cada ronda se muestran los fallos que se han tenido en ella y, finalmente, cuando se termina la partida, se muestra un *canvas* final en el que se presentan la puntuación total y los fallos totales. Además, surge un botón que permite volver a jugar con la misma configuración.

El guardado de datos a formato CSV del que se ha hablado anteriormente se realiza con la finalización del juego, por lo que si la partida no termina se entenderá que no se quieren guardar los datos de esta.

Diagrama de flujo

Esta escena ha sido diseñada teniendo en cuenta una gran variedad de situaciones. Para facilitar la comprensión de estas se expondrán todas las operaciones en un diagrama de flujo.

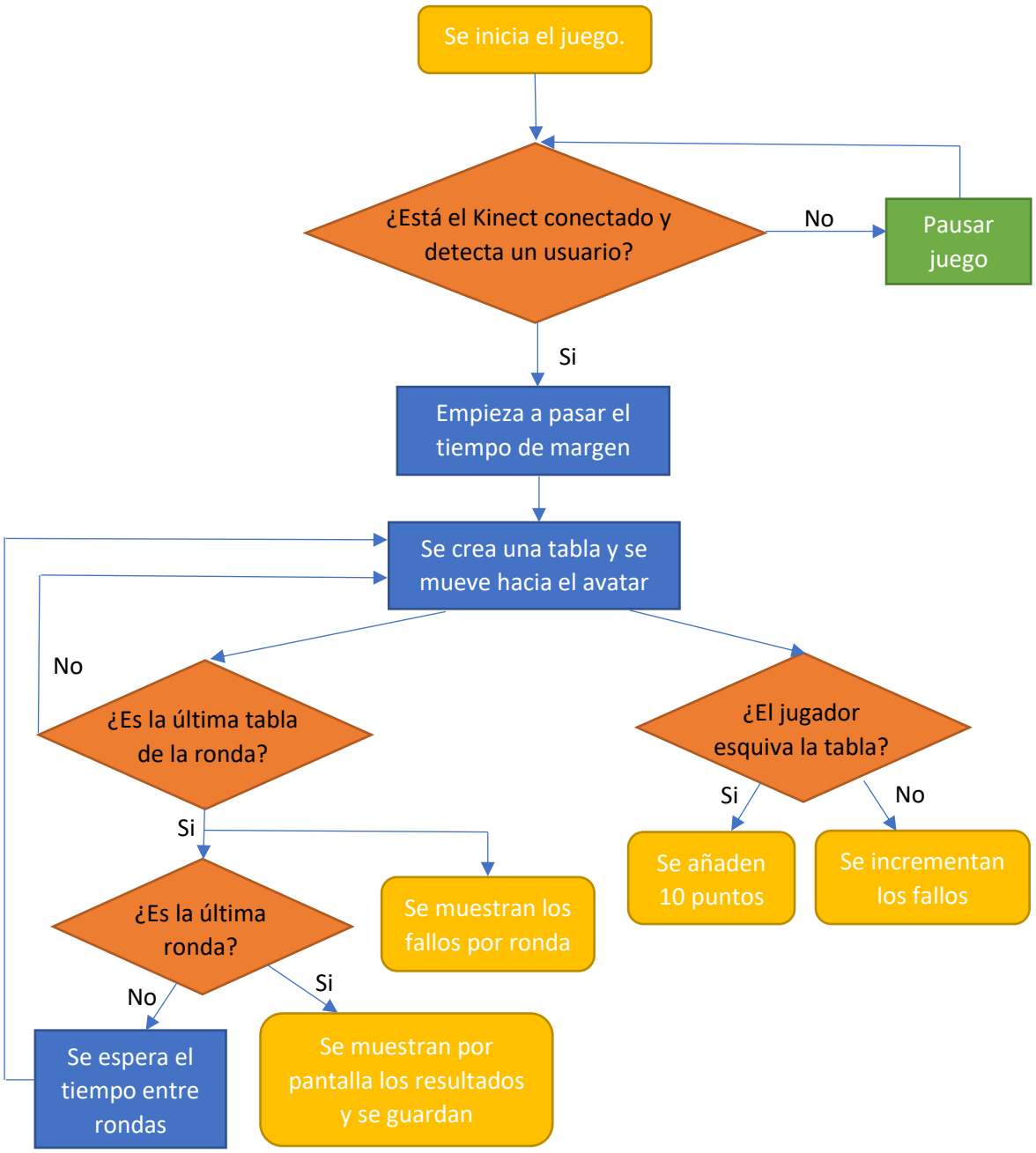

Figura 5.19: Diagrama de flujo del juego de esquivar

5.3.3 Evaluación mediante el test *Fugl-Meyer*

En este caso, la escena se ha desarrollado con el objetivo de aplicar la mayor cantidad posible de ítems correspondientes al test *Fugl-Meyer*. Debido a las características de estos, es imposible poder implementarlos todos. Algunos requieren que sea el terapeuta el que mueva al paciente, en otros se puntúa el dolor al realizar ciertos movimientos y otros ponen en riesgo la integridad del paciente si el terapeuta no aporta su ayuda. En el siguiente apartado, "Resultados", se expondrán detalladamente todas las decisiones tomadas para la elección de los diferentes movimientos.

Al igual que en la escena anterior, ésta también posee un botón de salida que permite volver al menú principal en cualquier momento. También comparte con el caso anterior la pantalla de espera, la cual pausa el juego hasta que un jugador haya sido detectado, y el margen de tiempo otorgado una vez que se ha detectado un usuario, para que este tenga tiempo suficiente para prepararse. En este caso, sin embargo, será de tres segundos.

En cuanto a los parámetros que permiten ser modificados, solo se puede variar el tiempo máximo que se le permite al paciente estar intentando realizar un movimiento. Este tiempo es determinado por el terapeuta en el menú de opciones. Si no se selecciona ninguno se trabajará con el de la sesión anterior.

Debido a motivos de precisión, solo se puede jugar con el avatar femenino en esta escena, ya que los *colliders* de este avatar han sido a los que se les ha dedicado más trabajo y, por tanto, son los más precisos y los que inducirán menos error en los movimientos.

Figura 5.20: Captura de la evaluación mediante el test *Fugl-Meyer*.

Objetivo y mecánicas de juego

Como ya se ha explicado en numerosas ocasiones, el objetivo de este juego es el de evaluar al paciente siguiendo el test *Fugl-Meyer*. Para ello, se le indicará al paciente que vaya realizando diferentes ejercicios, los cuales se puntuarán individualmente, y tendrán un tiempo máximo para su finalización.

Para indicar el movimiento a ejecutar, se emitirán señales tanto auditivas como visuales. Esto quiere decir, que antes de realizar cada movimiento, se mostrará un video en el que se muestra al avatar realizando el susodicho movimiento. Además, se emitirá una pequeña nota de voz en la que se dirá el nombre del movimiento a realizar. Por último, se mostrará en pantalla el nombre del movimiento. De esta forma, se pretende que el paciente pueda recibir el estímulo de la mayor cantidad de maneras posibles, lo cual puede ser beneficioso para aquellos que tengan algún tipo de patología relacionada con la recepción de estímulos externos.

Una vez se hayan terminado todas estas señales, el paciente recibirá una señal para que pueda empezar el movimiento. Esta señal será tanto visual, como auditiva. Se mostrará un mensaje por pantalla, que será "Comienza", y un audio con el mismo comunicado. A partir de este instante, se medirá el tiempo que tarda el usuario en finalizar. Así, se puede analizar los datos una vez que ya se ha recibido el estímulo, pudiéndose detectar el tiempo de reacción y el tiempo empleado en cada movimiento. De esta forma, se tiene un mayor control sobre el estado y la progresión del paciente, permitiendo una mejor planificación de su terapia.

Tras finalizar con todos los ítems aparecerá un mensaje final en el que se le indicará al usuario que ha terminado. Junto a este mensaje se ha situado un botón que permite volver a jugar con los mismos parámetros. Además, para favorecer la motivación del paciente, se muestra la puntuación que se ha obtenido, consiguiéndose así que este sea consciente de su progreso.

Programación

Esta escena también se ha realizado con *colliders*, ya que se ha considerado el mejor método para evaluar que el movimiento del paciente se ajuste al que debería ser. Los *colliders* asignados al avatar son los mismos que en el caso anterior, mientras que los de los distintos movimientos deben tener activada la opción de *is trigger*. Es necesario destacar que los movimientos no necesitan el componente *rigidbody*, mientras que el personaje si.

El avatar está compuesto por tantos *colliders* como articulaciones se requieren. Es decir, existe uno para la mano, otro para el antebrazo, otro para la parte superior del brazo etc. En la **figura 5.21,** que muestra un ejemplo de movimiento, también se pueden apreciar los tres *colliders* con forma de prisma circular que forman el brazo.

Los movimientos se han creado a partir de diferentes *Gameobjects*. A estos se les ha eliminado la maya de renderizado, de forma que lo único que contuviesen fuese el componente de *collider*. A partir de ahí se han ido creando diferentes configuraciones para cada ítem. No obstante, todas tienen ciertos elementos en común.

- *Goal*. Este *Gameobject* tiene forma de plano y se coloca en la parte final del movimiento. Tiene adjunto un *script* que permite detectar *colliders*, de forma que cuando todos los necesarios hagan contacto con él, se sabrá que el movimiento se ha realizado correctamente.
- *Espacio permitido*. Este es el volumen en el que el paciente debería de realizar el movimiento. Principalmente, se ha diseñado por si alguno sufre de fuertes temblores o trayectorias poco efectivas. Funciona detectando los *colliders* que salgan de este espacio. A partir de cierta cantidad, el movimiento es penalizado y solo permite obtener una puntuación máxima de un punto, ya que se considera que, aunque llegue a la posición final, se ha realizado con dificultades.
- *Goal_intermedio*. Es un componente exactamente igual a *Goal*. Sin embargo, existe una gran diferencia. Este se coloca a mitad del recorrido, si el paciente es capaz de alcanzar ese punto antes del tiempo límite asignado, se le da un punto, ya que, a pesar de no realizarlo por completo, ha conseguido hacerlo parcialmente.

Estos elementos se pueden observar en la siguiente imagen.

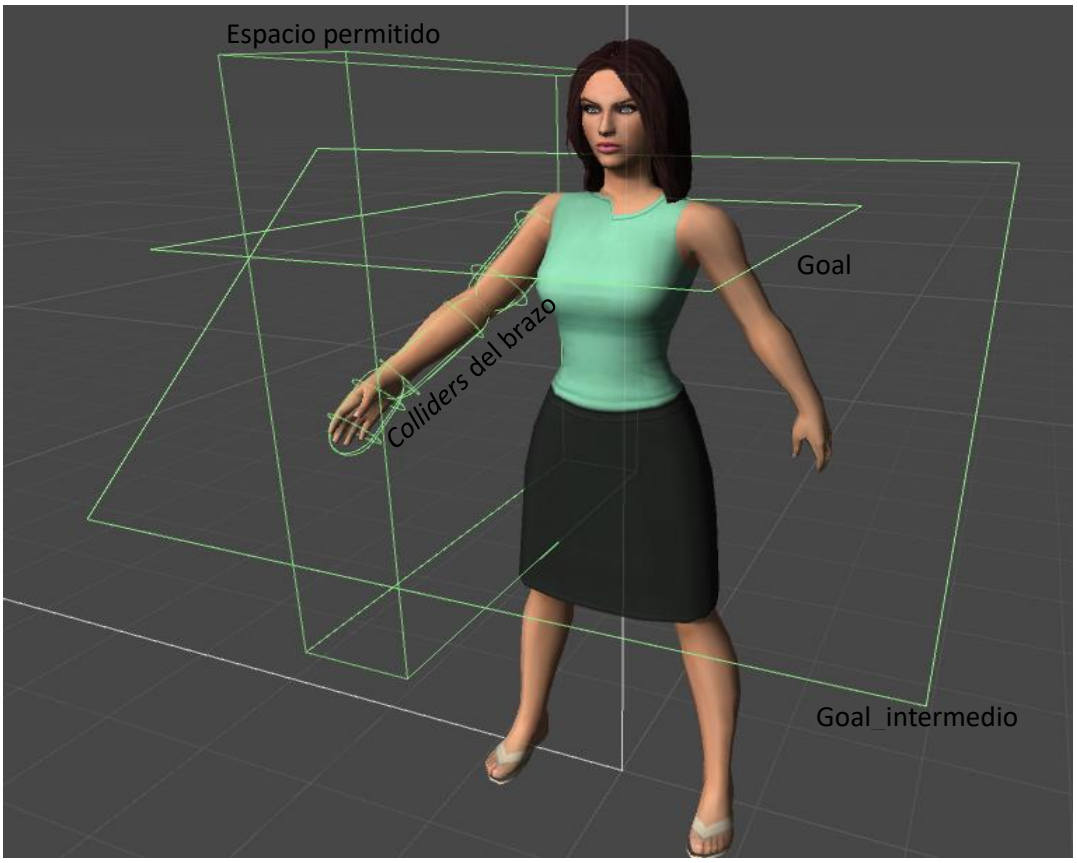

Figura 5.21: *Collider* que definen el movimiento

Existen otros movimientos que no se pueden detectar solo usando estos elementos, por lo que es necesario añadir más. Estos nuevos elementos son *goal inicial* y *goal mantener*. Ambos tienen las mismas bases que el componente *goal*. El primero de ellos es prácticamente idéntico, con la particularidad de que no otorga ningún punto,

simplemente obliga al paciente a que la posición inicial sea la correcta. Si este elemento no detecta que la posición inicial es correcta no se dará ningún punto en el ejercicio. El segundo es similar, no obstante, varía en que la posición que detecta ha de mantenerse, es decir, no solo vale con que esa sea la posición inicial, sino que se tiene que mantener durante la realización de todo el movimiento.

En la **figura 5.22** se muestra la "rotación externa del hombro derecho". En este ejercicio, la parte superior del brazo derecho tiene que estar siempre en contacto con *goal mantener*. Además, esta parte, junto con el antebrazo y su mano, no puede salir del espacio permitido. Teniendo esto en cuenta, si los tres elementos consiguen llegar a contactar con *goal intermedio* o *goal* se obtendrán uno o dos puntos respectivamente.

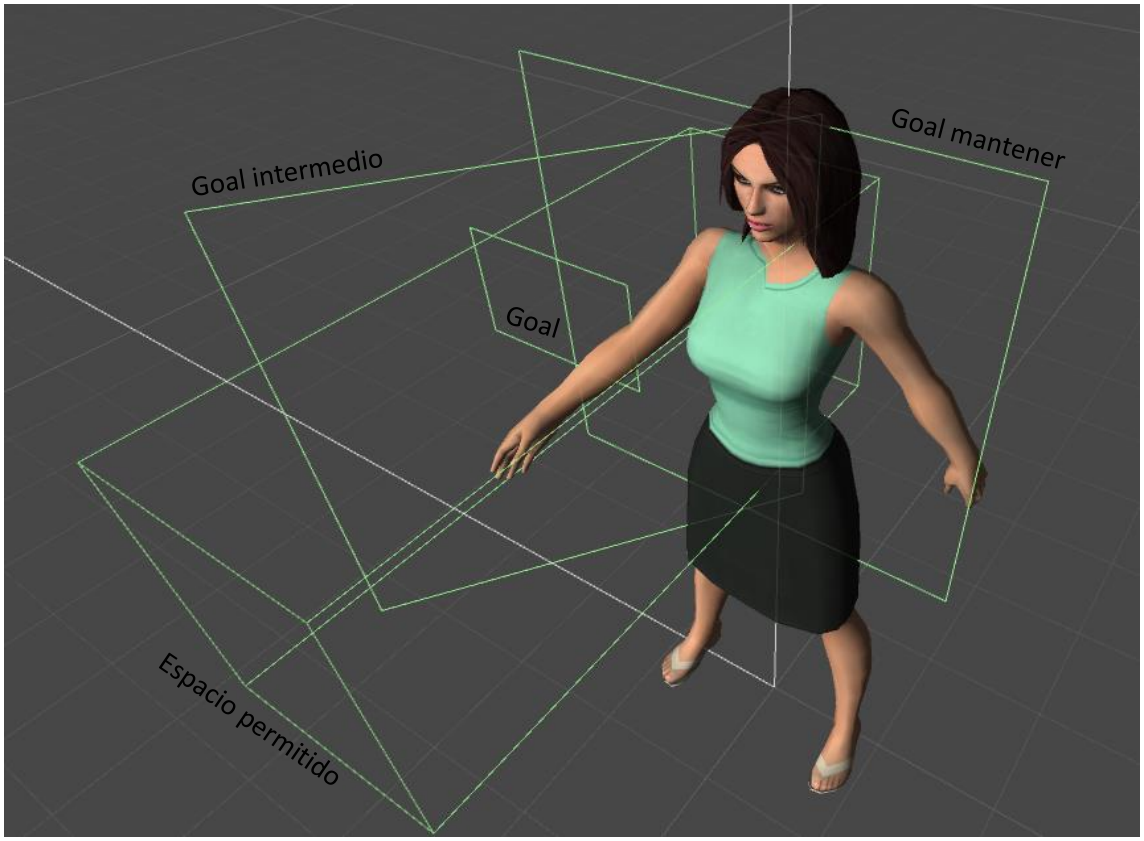

Figura 5.22 Muestra de todos los *colliders* usados para definir un ítem

Una vez realizados todos estos *script*s, ya es posible realizar todos los ejercicios. No obstante, aún hay que englobar todos los *colliders* que formen cada movimiento como hijos de un *Gameobject*. De modo que cada uno de estos padres pueda acceder a todos los elementos que forman el movimiento y pueda determinar así la puntuación obtenida. Además, todos estos serán hijos de otro *Gameobject*, llamado *Movimientos*. Este se encarga de gestionar todo lo relacionado con los ítems. Sus principales funciones son recoger todos los datos útiles y determinar cuándo se activa y desactiva cada movimiento. Debido a la propia naturaleza de los *colliders*, estos han de desactivarse cuando su movimiento no se esté realizando, puesto que podrían estropear el resultado.

Cuando se terminen todos los ítems se guardarán la puntuación y el tiempo, tanto por ejercicio, como de forma global, en un archivo CSV. Esto implica que, si se interrumpe el juego durante la sesión, no se guardarán los resultados.

Diagrama de flujo

Al igual que en el caso anterior, esta escena está compuesta por una gran cantidad de *scripts*. Por tanto, se pueden dar una gran cantidad de situaciones. Para mejorar la comprensión de la globalidad del juego se mostrarán los procesos en un diagrama de flujo.

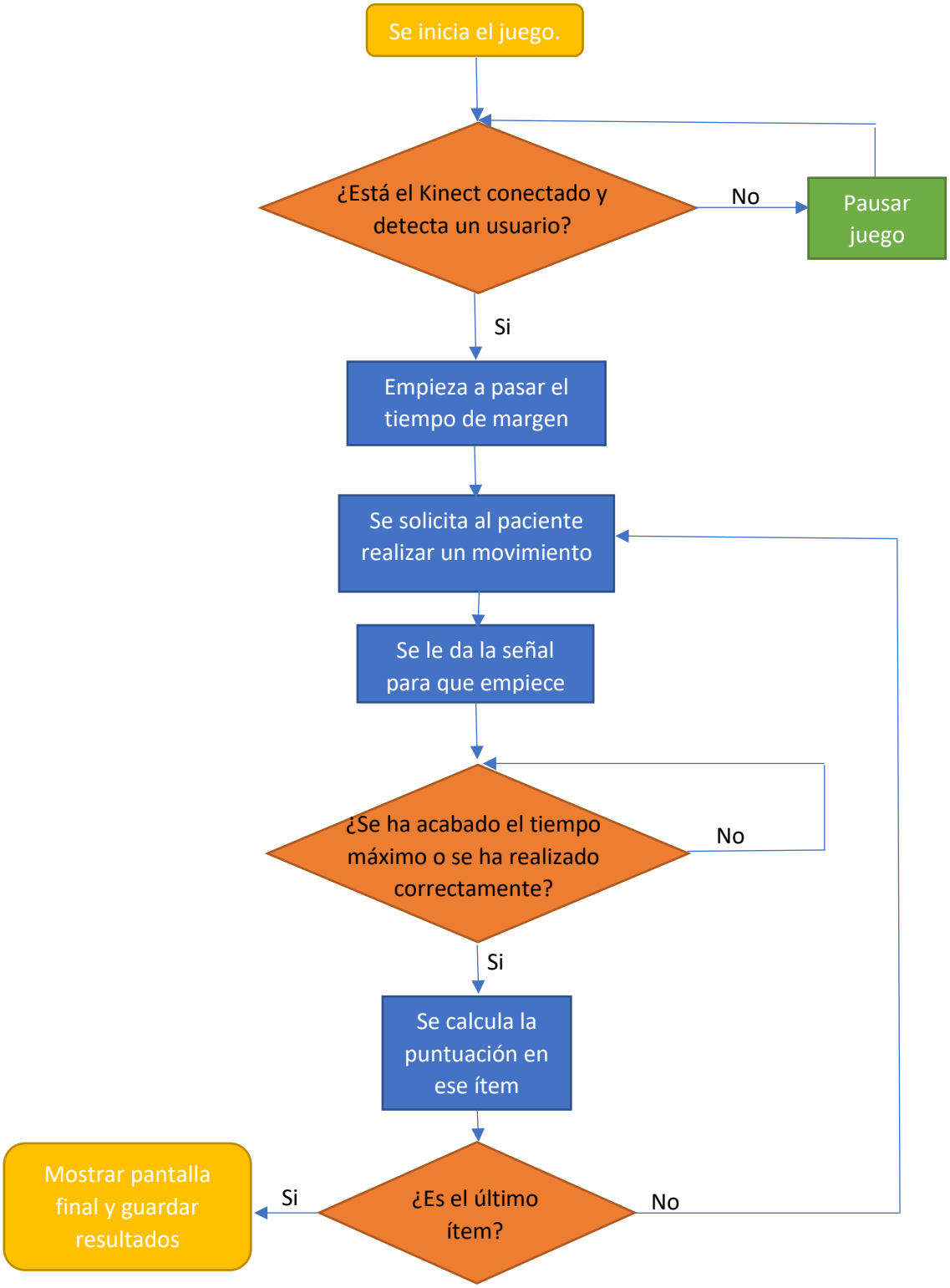

Figura 5.23: Diagrama de flujo de la evaluación mediante *Fugl-Meyer*

6 Resultados

Llegados a este punto, solo queda explicar el porqué de los ítems escogidos para la aplicación. Para ello, es necesario exponer en detalle las principales razones por las que se han descartado ciertos movimientos, mientras que se han elegido otros. También es importante que se razonen las distintas formas de realizar cada uno, esto es, la distancia al sensor y el plano en el que trabajar. Para llegar a estas conclusiones se han realizado una serie de pruebas para valorar la precisión del *Kinect*. Estas se expondrán en este punto.

6.1 Experimentos

Según un estudio realizado en 2015 sobre el *Kinect 2.0* [36], la distancia ideal a la que debe estar el usuario respecto al sensor es de entre 1 y 2.5 metros, ya que es la distancia en la que se obtienen las mediciones más precisas. En nuestro caso, se pedirá una distancia de entre 1.8 y 2 metros, puesto que una menor podría provocar que los pacientes altos se salieran del campo de visión del *Kinect* al realizar ciertos movimientos y una mayor estaría demasiado cerca del límite de medición óptima.

Otro estudio, realizado en el centro de investigación alemán de inteligencia artificial, compara las dos versiones existentes de *Kinect* en el mercado [37]. Los resultados obtenidos indican que el *Kinect 2.0* tienen un offset medio de -18mm. Este valor oscila con la distancia al sensor, pero en términos generales se mantiene constante. No se da el mismo caso con la primera versión de *Kinect*, la cual, a distancias pequeñas, tiene un offset menor a 10mm. Sin embargo, este aumenta con la distancia llegándose a un valor de hasta 40mm. Es obvio que la segunda versión es preferible en este apartado, no solo por ser un offset medio menor, sino porque al ser constante es mucho más sencillo de eliminar.

En adición a estos estudios, se realizaron pruebas comparativas entre el sensor *Kinect 2.0* y el *Vicon* para poder observar los planos en los que había menos similitudes. Se conoce con certeza que el segundo sistema realiza un seguimiento del usuario muy preciso, por lo que se toma de referencia. Si el *Kinect* no registra los mismos parámetros, se considerará que la variación es un error en la medida.

Tras realizar varios movimientos en los planos anatómicos, mostrados en la **figura 6.1**, se comprobó que el plano sagital es el que posee un mayor error en la medida. Esto es así si el *Kinect* está situado frente al usuario, puesto que, al realizar movimientos a través del susodicho plano, las articulaciones se solapan desde el punto de vista del sensor, lo que complica su detección. También surgieron problemas en ciertos puntos del plano transversal, concretamente en las zonas más cercanas al plano sagital. Siendo provocados por el mismo motivo. Por tanto, se puede concluir con que, si el *Kinect* se encuentra frente al usuario, los movimientos que impliquen un desplazamiento en el plano sagital pueden provocar errores, siempre y cuando pueda existir solapamiento de *joints.*

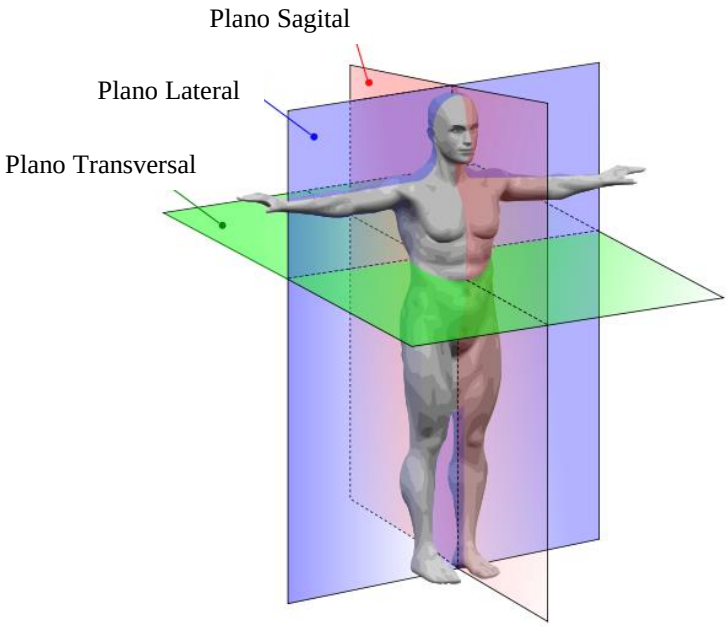

Figura 6.1: Planos anatómicos

Para corroborar todos los datos mencionados hasta ahora e intentar añadir otros nuevos se realizaron una serie de mediciones en el laboratorio. Estas, además, se usan para determinar que ítems del test *Fugl-Meyer* se pueden llevar a cabo y cuales son necesarios descartar. Las pruebas analizan la precisión en la medida de posición y ángulo, los planos en los que se realizan mejores medidas y los problemas que pueden surgir al utilizar el Kinect en ciertas condiciones, como poca luz o con ropas anchas.

En primer lugar, se llevó a cabo una evaluación del error que podría tener el sensor al detectar la posición de las articulaciones. La primera prueba realizada consiste en medir una distancia verticalmente desde el suelo. En el punto obtenido se realiza una marca, o cualquier señal que permite mantener esa posición de forma exacta. A continuación, se sitúa una articulación que pueda detectar el *Kinect* en ese lugar y se toman varias medidas. Es importante que durante el proceso no haya modificaciones ni en el Kinect ni en la marca, para evitar así tener errores debidos a la variación de los parámetros iniciales.

Cabe remarcar que las medidas tomadas están ajustadas al número más grande que no oscilaba. Esto quiere decir que, durante las pruebas, el programa mostraba las medidas con 6 decimales, sin embargo, estos oscilaban muy fácilmente. Por tanto, el número tomado se corresponde con las cifras estáticas, que no sufrían ningún cambio. Estas variaciones se pueden deber a dos motivos: una falta de sensibilidad en el sensor para percibir cambios tan pequeños o una falta de estabilidad en la persona de la cual se toman las mediciones, ya que cualquier mínimo movimiento puede hacer cambiar esos decimales.

Los resultados obtenidos son los siguientes:

Articulación	Altura de la marca (m)	Medidas tomadas (m)	Media de las medidas
Muñeca derecha	1.10	1.08	1.086
		1.075	
		1.12	
		1.08	
		1.09	
		1.072	
Muñeca izquierda	1.10	1.086	1.091
		1.10	
		1.11	
		1.09	
		1.07	
		1.091	
Codo derecho	1.40	1.42	1.411
		1.41	
		1.417	
		1.40	
		1.41	
		1.414	
Codo izquierdo	1.40	1.42	1.414
		1.412	
		1.422	
		1.40	
		1.41	
		1.42	

Tabla 6.1: Primer experimento

En este primer experimento se han realizado medidas con ambos lados del cuerpo. Como se puede apreciar, no hay diferencias apreciables entre los resultados del lado derecho e izquierdo, por lo que se podría afirmar que la precisión de las medidas es indiferente del lado.

Se puede observar como en muy pocas ocasiones la medición se corresponde con la medida real. Sin embargo, el error no es mayor a 2 cm en ninguna ocasión. Además, se puede apreciar cierta uniformidad en las medidas, esto quiere decir que, aunque tengan un error respecto a la real, entre todas ellas son bastante constantes. Esto es un aspecto muy positivo si se quieren comparar las posiciones de dos articulaciones, ya que para ello el valor real es indiferente.

Por último, se puede apreciar como en el caso de la muñeca la mayoría de las medidas se sitúan por debajo de la real, esto puede deberse al offset del que se ha hablado anteriormente. No obstante, en el codo, están algo por encima. A falta de más pruebas se puede concluir con que el offset puede depender de la articulación o simplemente ha habido un error humano a la hora de situar el codo en la posición correcta.

La segunda prueba es similar a la anterior. En este caso, se ha medido una distancia de 1.90 metros verticalmente desde el suelo y se han realizado marcas cada 10 cm. Con esto se pretende analizar si la altura de la medición tiene impacto en el error de la medida.

En este caso se han tomado medidas de la muñeca derecha, las cuales son:

Altura real(m)	Medida obtenida 1 (m)	Medida obtenida 2 (m)
1	1.04	1.00
1.10	1.12	1.10
1.20	1.21	1.19
1.30	1.31	1.29
1.40	1.404	1.40
1.50	1.51	1.507
1.60	1.60	1.60
1.70	1.702	1.69
1.80	1.79	1.79
1.90	1.90	1.88

Tabla 6.2: Segundo experimento

Como se puede observar en la tabla hay una cierta tendencia, en el primer caso, las medidas tienen un error inicial muy grande que va disminuyendo según se incrementa la altura, mientras en el segundo, el error comienza en cero y va aumentando con la altura. Esto se podría traducir como que el sensor es preciso en un rango de altura, es decir, si se calibra para que en 1.90m sea exacto habrá un error al llegar a 1m, mientras que, si se calibra para 1m, el error aumentará al acercarse a 1.90m. Por tanto, el sensor será más preciso en las distancias más próximas a la zona calibrada, según estas aumente el error en la medida aumentará. Aparte de esto, se puede considerar como que los resultados son los suficientemente exactos, ya que el error no es muy grande.

El siguiente experimento consiste en evaluar el método que se va a usar para medir los ángulos. Para ello, se marcan distintos ángulos con un transportador. A continuación, se sitúan las articulaciones a evaluar con esos ángulos y se obtienen las mediciones del sensor. En esta ocasión, se ha realizado la prueba en el plano lateral con el ángulo que forma la línea que atraviesa la muñeca y el hombro derecho con la vertical, es decir, elevando el brazo lateralmente

Angulo real (º)	Angulo obtenido 1 (º)	Angulo obtenido 2 (º)
0	0.6	1.8
15	14	16.4
30	30.4	31.3
45	45.9	43.2
60	61.6	59.7
75	74.3	75.5
90	88	91.2

Tabla 6.3: Tercer experimento

En esta ocasión hay que recordar que el método usa las posiciones de las articulaciones para calcular los ángulos, por lo que el error de ambas posiciones se sumará. Esto se puede apreciar directamente en la cantidad de números que oscilan, ya que en esta prueba solo se ha mantenido fijo, a lo máximo, un decimal.

A pesar de ello, se pueden considerar unos resultados más que adecuados, puesto que el error absoluto nunca es mayor a 2º. Es más, en la mayoría de medidas no supera el grado de desviación. En este caso, no se aprecia ninguna tendencia, los errores surgen a lo largo de toda la prueba.

Posteriormente, se realizó la misma prueba, pero en esta ocasión con el ángulo que forman el cuello y la parte baja de la columna vertebral con la vertical en el plano sagital, es decir, inclinando el cuerpo hacia delante. Las medidas obtenidas son las siguientes:

Angulo real (º)	Angulo obtenido 1 (º)	Angulo obtenido 2 (º)
0	1.4	1.5
10	11.4	10.8
20	17.9	19.2
30	29	30.4
40	37	39.3
50	44.2	48
60	51	54
70	64	59

Tabla 6.4: Cuarto experimento

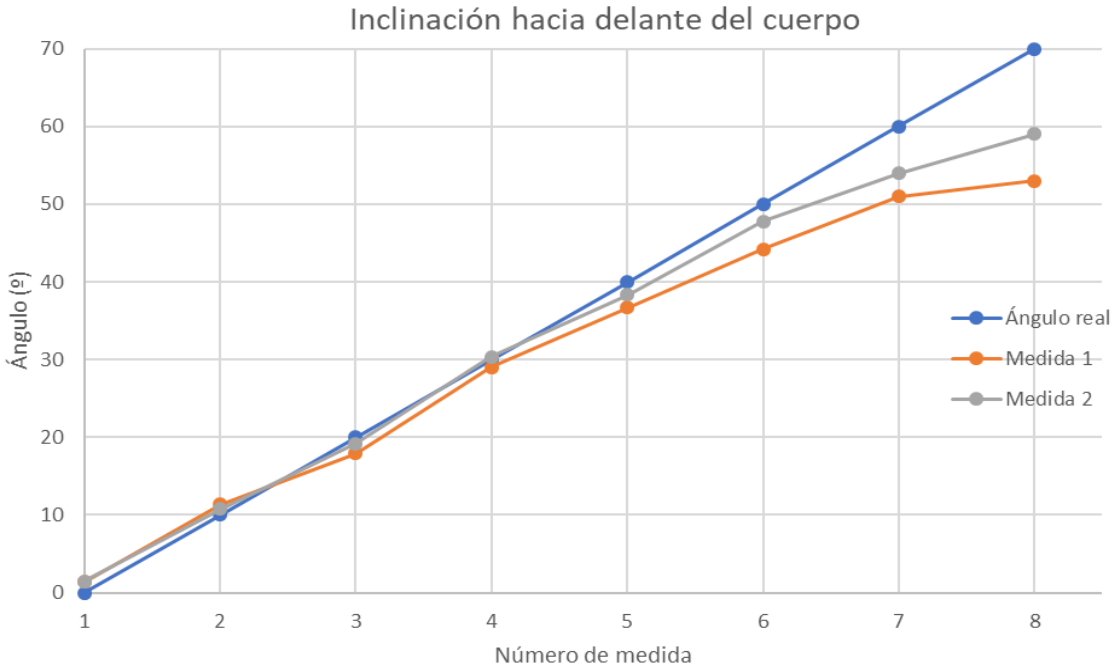

Figura 6.2: Gráfica de resultados del cuarto experimento

En esta ocasión se están realizando medidas en el plano sagital, por lo que se espera un aumento drástico del error al llegar a posiciones en las que se solapen articulaciones. Y, en efecto, esto sucede aproximadamente a partir de los 40º, siendo totalmente evidente a partir de los 50º. En la **figura 6.2** se puede apreciar como el error se dispara a partir de los 40-50º. Si se ignora esto, las demás medidas son considerablemente válidas, ya que se mantienen alrededor del ángulo real, con un error absoluto menor a 2º, el cual es aceptable para el movimiento que se está realizando.

En la siguiente prueba, se evaluará la precisión al medir uno de los ítems del *Fugl-Meyer*. Se realizará una elevación frontal del hombro derecho, tanto de frente como de perfil al sensor, para evaluar así la diferencia entre realizarlo en un plano u otro. Se tomará el ángulo en una posición relajada y en la final. Al igual que en casos anteriores, la línea imaginaria del brazo estará formada entre la muñeca y el hombro.

De frente				**De Perfil**			
Derecho		**Izquierdo**		**Derecho**		**Izquierdo**	
Inicio	Final	Inicio	Final	Inicio	Final	Inicio	Final
12.7	91.8	14.8	96.1	13	90.4	14.1	91.5
10.3	88.4	13.7	93.4	14.6	91	13.4	90.6
12.5	90.5	13.8	94.2	14.1	91.1	15.4	92
11.4	92.4	12.1	92.2	15.1	91.7	13.5	90.1
11.8	91.6	12.8	92.3	13.8	90.2	14.6	91.7

Tabla 6.5: Quinto experimento

Lo más destacable a simple vista es que la posición inicial no es de 0º en ninguna ocasión, sino que empieza entre 10º y 15º. Esto se debe a que durante una posición relajada el codo no está totalmente recto, ni el brazo vertical hacia abajo, por tanto, al realizar la línea entre la muñeca y el hombro, está ya está en una posición superior a 0º. También cabe destacar que no hay diferencia significativa entre el lado derecho y el izquierdo.

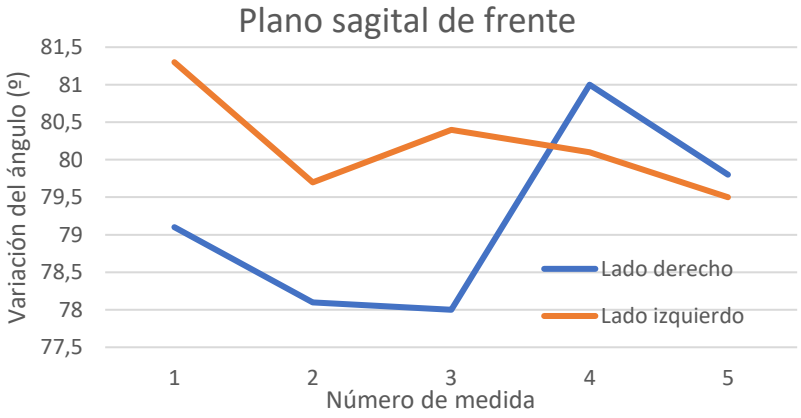

Figura 6.3: Resultados plano sagital de frente

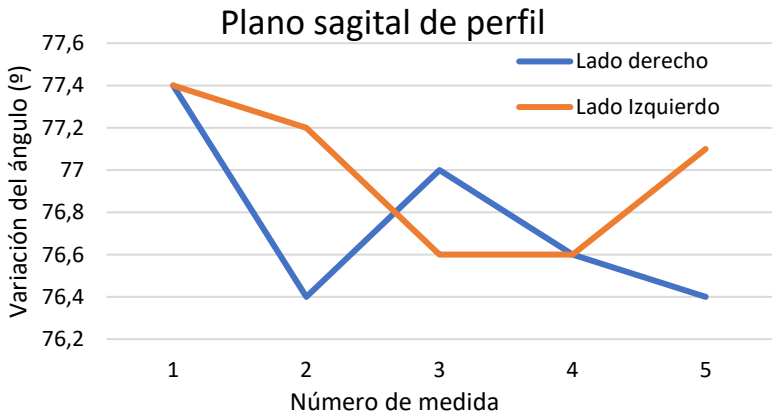

Figura 6.4: Resultados plano sagital de perfil

En cuanto a la diferencia entre una evaluación de frente o de perfil del usuario. Si se observan las tablas anteriores se puede apreciar como de perfil la variación del ángulo oscila solo 1º a lo largo de todas las medidas, mientras que de frente varía más de 3º. Además, es cierto, que las medidas realizadas de frente oscilan más en zonas cercanas a los 90º (por ser el plano sagital), pero, aun así, la mejora apenas es significativa al hacerlo de perfil, ya que solo se consigue un error de unos 2º menor. A cambio, el avatar pierde totalmente la postura, y las demás articulaciones dejan de ser rastreadas. En la **figura 6.5** se puede observar como el brazo a evaluar sigue bien rastreado, pero las demás articulaciones se pierden. Por tanto, para el modelo final, este tipo de movimientos se harán de frente, la diferencia en la precisión es demasiado baja como para que merezca la pena sacar al jugador de una experiencia de juego más inmersiva.

Figura 6.5: Avatar fallando al colocarse el usuario de perfil

Una vez trabajado con un movimiento en el plano sagital, se realiza uno en el plano lateral, este es una elevación lateral del hombro derecho. Se realiza de la misma forma que el anterior dando como resultado lo siguiente:

De frente		**De perfil**	
Brazo Derecho		Brazo Derecho	
Inicio	Final	Inicio	Final
7.5	89.3	10.5	99.6
8.4	89.7	9.3	97.7
6.5	87.9	10.6	92.3
7.4	88.3	9.9	95.3
6.3	88.1	10.1	97.2

Tabla 6.6: Sexto experimento

En este caso el movimiento realizado de frente al sensor es mucho más preciso, ya que de perfil no solo se solaparían las articulaciones del brazo como en el caso anterior, sino que aparte de estas, se solaparían con todas las del torso. Por tanto, este movimiento se realizará de frente al sensor, lo cual, en adición, favorece a la inmersión en el juego, ya que el paciente no tiene que realizar ningún movimiento extra como podría ser el de

colocarse. Cabe destacar, que como en el caso anterior, el ángulo de inicio no es cero, puesto que el ángulo del brazo con la vertical no es cero naturalmente.

Esto es importante remarcarlo, ya que en la **figura 6.6** se puede observar como el ángulo recorrido cuando se realiza de perfil es significativamente mayor y se acerca más a 90º. En este caso, no es algo positivo. La variación correcta debe rondar alrededor de los 80º, puesto que como ya se ha explicado, en una posición relajada, el brazo no apunta directamente hacia abajo. Además, la medición es mucho más constante al realizarse de frente al sensor, lo cual confirma que es la elección correcta.

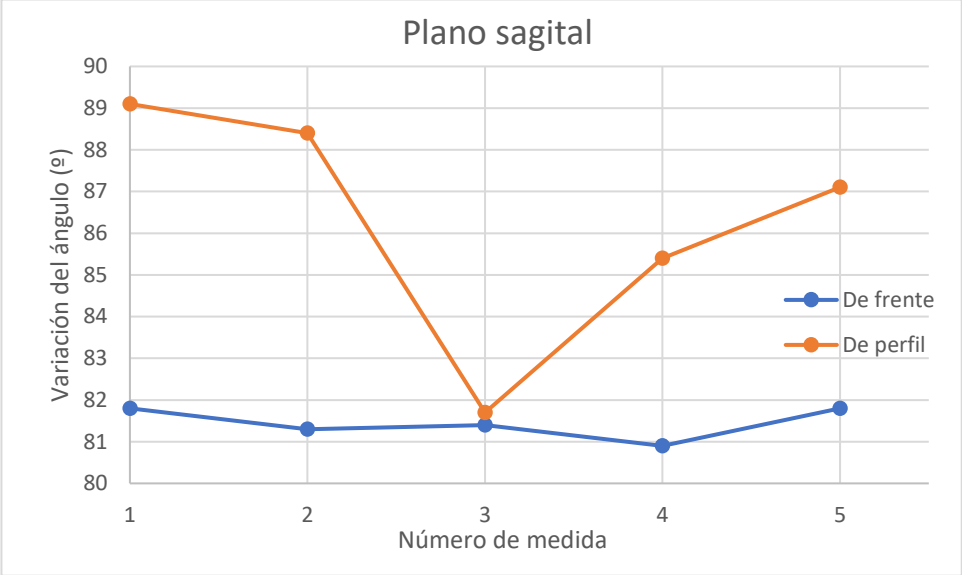

Figura 6.6: Resultados plano sagital

Ya que se han realizado un movimiento en el plano sagital y otro en el lateral, se va a efectuar otro en el plano transversal. Este consistirá en mover el brazo desde la posición final de una elevación lateral de hombro hasta la final de una elevación frontal de hombro. Es decir, el hombro a 90º y el codo a 0º, desde un lado hasta el frente.

Inicio	Final
158	83.9
158.7	87.2
165	93
161	90
160	88.5
158.2	87.4

Tabla 6.7: Séptimo experimento

Tal y como se muestra en **tabla 6.7**, la variación del ángulo no llega a ser 90º, sino que se queda en una media de 71. 81º, aproximadamente 20º por debajo. Esto se debe en parte al error humano al realizar el movimiento, pero en gran parte al sensor en la parte inicial. Como se puede ver en los resultados, esta se queda muy lejos de los 180º que debería de alcanzar, lo que nos indica que el sensor no medirá los movimientos en este plano eficazmente. El final también tiene cierto error, provocado por el momento en el que las articulaciones se solapan.

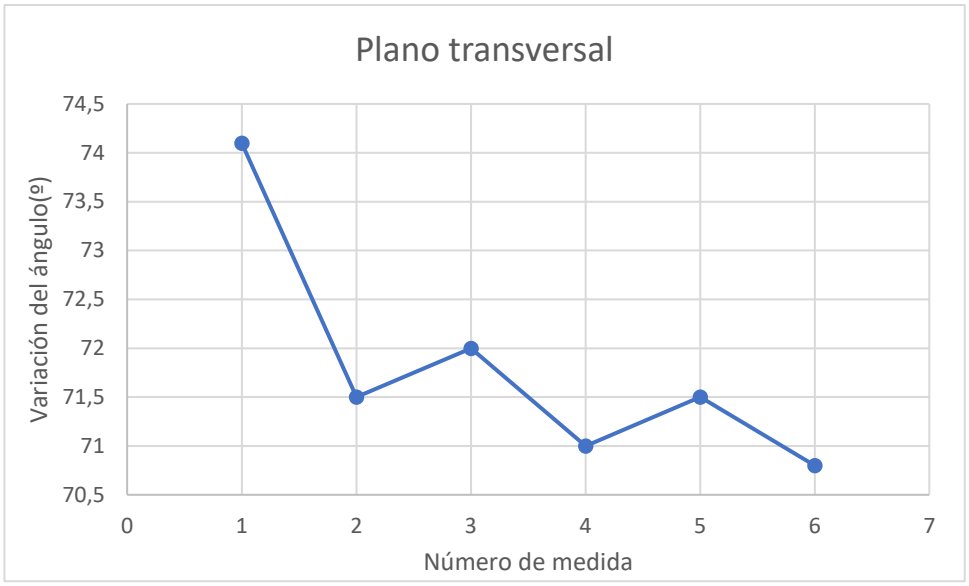

Figura 6.7: Resultados plano transversal

Si se analiza la **figura 6.7** se puede concluir con que la medida suelte ser bastante precisa, situándose la variación del ángulo recorrido en menos de 2º. Sin embargo, la primera medición no cumple esta equidad. Probablemente por un fallo en la medida, un movimiento mal realizado o un fallo puntual del sensor, ya que como se puede ver es un caso aislado.

Debido a estos resultados en el plano transversal, se quiso indagar más en este nuevo problema de precisión. Para ello, se realizó otro movimiento en este plano, de forma que se comprobaría si las medidas iniciales realmente son tan malas. Este movimiento consiste en una rotación del hombro, tanto interna como externa. El movimiento se muestra en la **figura 6.8**.

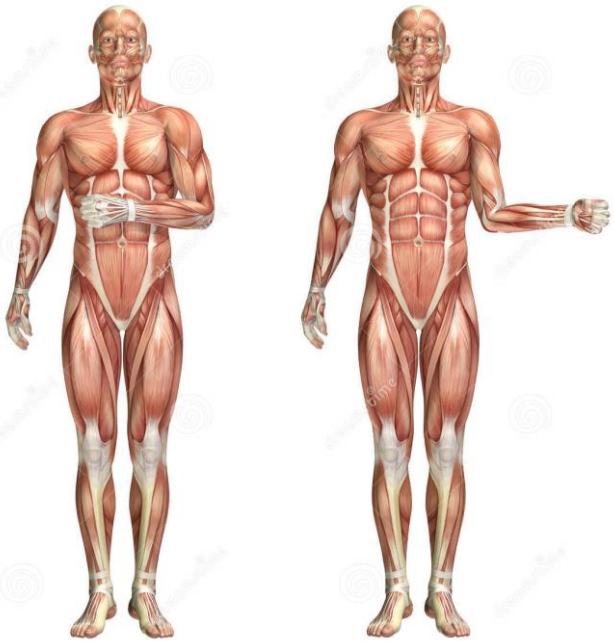

Figura 6.8: Rotación interna y externa del hombro

En ambos casos se han realizado pruebas de frente y de perfil al sensor.

Rotación externa				Rotación interna			
De frente		**De perfil**		**De frente**		**De perfil**	
Inicio	Final	Inicio	Final	Inicio	Final	Inicio	Final
82	152.4	-5.2	80	80	21.8	-10	-36
78	156.2	-4.6	84.3	76	18.6	-7	-28
85	155.7	-2.5	81.4	75	19.1	-8	-42
80	154.5	-5.3	82	78	18	-6	-24

Tabla 6.8: Octavo experimento

Antes de comentar los resultados es importante destacar que cuando se realiza de perfil, siempre se hace desde el lado más favorecido.

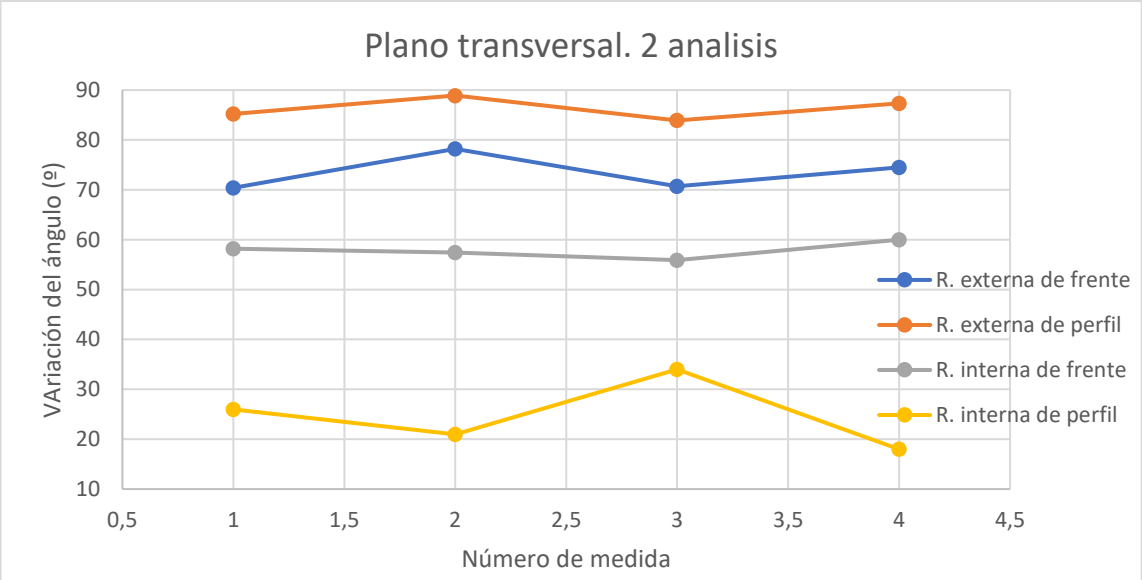

Figura 6.9: Resultado plano transversal. 2ªprueba

En cuanto a los resultados, se puede observar como en la rotación interna el realizarlo de perfil perjudica gravemente la medida, probablemente debido a que el propio brazo bloquea la visión del Kinect. Sin embargo, en la rotación externa no es así, por lo que se obtienen medidas mucho más exactas. En cuanto al rango de movimiento, se puede observar que se encuentra en unos 60º para la rotación interna y en unos 80º para la externa. Esto se debe a que la rotación interna se detiene cuando el brazo choca el cuerpo, por lo que será menor que la externa, la cual no suele llegar a 90º por falta de flexibilidad en el hombro. En este sentido, las dos pruebas realizadas de frente son mejores.

Si se comparan las medidas se puede observar como son más exactas en la parte final del recorrido al realizarlo de frente, y en la inicial al realizarlo de perfil. Esto se debe a los planos en los que mide mejor el Kinect, de los cuales ya se ha hablado. En nuestro proyecto es preferible que la medición final sea lo más exacta posible, ya que la posición inicial no suele ser difícil de alcanzar, mientras que la posición final es la que indica si el paciente ha realizado bien el ítem. Por tanto, este tipo de movimientos se realizarán de frente al *Kinect*.

Finalmente, podemos decir que las medidas realizadas en el plano transversal pueden ser consideradas válidas, siempre que se realicen de frente y aceptando que habrá un error mayor al acercarse al plano sagital.

En cuanto a uso del Kinect en condiciones de poca luz, este empeora en parte sus medidas. Sin embargo, a pesar de ello, no es un factor determinante. Algo que tiene más influencia es el tamaño de la ropa, si un usuario viste ropas extremadamente anchas, el sensor puede confundir la posición de ciertas articulaciones. Por lo tanto, ambos factores son importantes para tener en cuenta una vez se realicen las pruebas reales.

6.2 Ítems del *Fugl-Meyer*

Como ya se ha explicado, el *Fugl-Meyer* es un test que evalúa el grado de afectación de un ictus. Para ello, se usa una escala que va desde los cero hasta los dos puntos, equivaliendo un cero a que no se ha realizado el ítem, uno a una realización parcial y dos a una ejecución completa. La puntuación máxima es de 226 y viene determinada por el número de ítems, de los que hay 113. La división de esta puntuación se realiza de la siguiente forma: 100 puntos para la función motora, concretamente 66 para el tren superior del cuerpo y 34 para el inferior, 24 puntos para la sensibilidad, 14 para el equilibrio, de los cuales 6 son sentado y 8 de pie, 44 para el rango de movimiento de las articulaciones y los últimos 44 para el dolor.

Teniendo en cuenta los resultados y las limitaciones obtenidas en los experimentos y con el fin de integrar la mayor cantidad de ítems posibles, se procederá a analizar estos en detalle:

- Rango articular y dolor articular. Esta sección corresponde a los ítems comprendidos entre el 70 y el 113 incluidos. Estos ítems tratan de medir la movilidad pasiva del paciente y el grado de dolor que sufren. Por lo tanto, quedan descartados para incluirlos en el *Serious Game*, puesto que no se dispone de ningún método de mover las articulaciones del paciente sin la ayuda de una persona de forma correcta.
- Sensibilidad. Comprende desde el ítem 58 al 69, ambos incluidos. En esta ocasión, se trata de valorar la sensibilidad del paciente en distintas partes del cuerpo. Dado que no es posible tocar al paciente, este grupo también queda descartado.
- Valoración del equilibrio. Esta sección abarca los ítems del 51 al 57, ambos incluidos. Estos ítems comprenden movimientos que pueden poner en riesgo la integridad del paciente. En todos ellos existe la posibilidad de que el paciente no pueda mantener su estabilidad, por lo que para no sufrir ningún accidente no se llevarán a cabo.
- Valoración de la función motora en el miembro inferior. Este grupo integra todos los ítems desde la posición 34 hasta la 50, ambas incluidas. Debido a que algunos de estos movimientos pueden realizarse por el paciente de forma

independiente y a que no se pone en riesgo su seguridad, vale la pena analizar estos ítems más en profundidad.

- Actividad Refleja (Ítems 34 y 35). Como su propio nombre indica, estos evalúan los reflejos del paciente, por lo que no se pueden llevar a cabo en el proyecto.
- Sinergia flexora (Ítems 36, 37 y38) y sinergia extensora (Ítems 39, 40, 40 y 42). En estos el paciente ha de estar en una posición decúbito supino y decúbito lateral. Debido a que no se conoce si el paciente va a poder alcanzar esa posición sin ayuda y de forma segura se opta por no incluir estos ejercicios.
- Movimientos combinando sinergias (Ítems 43 y 44). En este caso, el paciente ha de estar en una posición sentada. En el ítem 43 se evalúa la flexión de rodilla a más de 90º. Al desplazar la pierna hacia atrás, el sensor deja de detectar la articulación, por lo que una medida de este ítem sería demasiado imprecisa. En cuanto al ítem 44, este evalúa la dorsiflexión de tobillo. Debido a que este movimiento posee un rango demasiado pequeño, el error que posee el *Kinect* es excesivamente grande como para obtener resultados fiables. Por tanto, ninguno de estos dos se implementará.
- Movimientos sin sinergias (Ítems 45 y 46). Ambos se realizan de pie. En el primero hay que realizar una flexión de rodilla y en el segundo una dorsiflexión de tobillo. El primer movimiento presenta unas condiciones adecuadas, por lo que se intentará su implementación, no como en el segundo, en el cual se requiere una precisión demasiado alta.
- Reflejos normales. Se corresponde con el ítem 47 y no puede ser incluido, ya que no hay forma de analizar los reflejos con el sensor.
- Coordinación (Ítems 48, 49 y 50). Estos tres ítems se puntúan con el mismo ejercicio, el cual consiste en poner el talón de un tobillo en la rodilla contraria y recorrer la tibia tan rápido como sea posible. Es necesario hacer 5 repeticiones seguidas. El ítem 48 puntúa el temblor, el 49 la dismetría y el 50 la velocidad. Esto significa que solo se podría implementar el 50, puesto que los otros requieren una precisión que no se tiene.

- Valoración de la función motora en el miembro superior. Es el último grupo de movimiento, y agrupa los Ítems 1 al 33. Estos movimientos tienen una alta probabilidad de ser implementados por lo que se analizarán en detalle.
  - Reflejos osteotendinosos (Ítems 1 y 2). No se pueden implementar por estar analizándose reflejos, como ya se ha explicado.
  - Sinergia flexora (Ítems del 3 al 8). Este conjunto evalúa la elevación escapular, la retracción, abducción y rotación externa del hombro, la flexión del codo y la supinación del antebrazo. Varios de estos movimientos son demasiados sutiles como para ser detectados adecuadamente. Los únicos que cumplen las condiciones para ser incorporados a la aplicación son el 6 y el 7. Estos serán implementados en el *Serious Game* final.
  - Sinergia extensora (Ítems 9, 10 y 11). En este caso las acciones a analizar son la rotación interna del hombro, la extensión del codo y la pronación

del antebrazo. Al igual que en el caso anterior, excepto el último, los demás serán implementados, ya que son movimientos extensos en recorrido que no requieren de afinar en gran cantidad la medida.

- Movimiento combinando sinergias (Ítems 12, 13 y 14). El ítem 12 consiste en mover la mano hacia la zona lumbar. Debido a que el sensor pierde la visión en esta zona no se sabría si el paciente lo ha realizado correctamente. El ítem 13 consiste en una flexión del hombro a 90º, con el codo a 0º, siendo un movimiento posible de implementar. El 14 es igual que el 13, con la diferencia de que el paciente debe de realizar una pronación y supinación del antebrazo. Recordemos que estos movimientos solo son detectados por el sensor si el dedo pulgar está separado, pero, puesto que el puño esté cerrado es condición del ítem, no se podrá implementar. Por tanto, este y el 12 no se implementarán.
- Movimientos que no combinan sinergias (Ítems 15, 16 y 17). El primero de estos consiste en una abducción de hombro a 90º, con el codo a 0º y el antebrazo pronado. El segundo consiste en mover el hombro de 90º a 180º, con el codo a 0º. El último consiste en una pronación y supinación del antebrazo mientras se mantiene una flexión en el hombro de unos 30-90 grados. En este grupo, los dos primeros ítems son muy favorables a medidas con el sensor, mientras que el tercero requiere de cualidades que no se poseen.
- Reflejos normales (Ítem 18). Por motivos ya explicados se ha de descartar.
- Muñeca (Ítems del 19 al 23). Todos estos requieren de un error mucho menor del obtenido para adquirir resultados fiables, por lo que no se implementarán.
- Mano (Ítems del 24 al 30). En estos ítems se realizan movimientos con los dedos, que el Kinect no detecta. Al igual que otros de agarre, los cuales no se pueden implementar sin un terapeuta que los realice.
- Coordinación (Ítems 31, 32 y 33). El paciente ha de llevar el dedo de la rodilla a la nariz lo más rápido posible. Al igual que en los ítems equivalentes del tren inferior, el único que se podría implementar es el 33, que mide la velocidad.

Una vez hecha la primera criba, en la que se han descartado los movimientos que de forma obvia no podrían ser implementaos, es hora de realizar una segunda, en la que se decidan cuáles serán los ítems finalmente implementados. Para ello, se realizarán todos los seleccionados hasta ahora, y dependiendo del porcentaje de acierto que se obtenga se incluirán en el *Serious Game* o no. Esto quiere decir, que, si se detectan correctamente más del 90% de las veces, serán considerados lo suficientemente exactos, mientras que, si no superan este porcentaje, el error sería demasiado grande como para considerarse fiables. De esta forma, se han realizado todos los ítems seleccionados 20 veces, apuntando en cada ocasión si la detección era correcta o no. De esta forma se han obtenido una serie de porcentajes que indican la fiabilidad de cada uno.

En primer lugar, los ítems 13 y 15 alcanzan sobradamente los requerimientos necesarios, ya que ambos han tenido un porcentaje de acierto del 100%. En cuanto al 6 y al 9, estos han tenido un porcentaje de 95 y 100% respectivamente. Finalmente, el 7 y el

10 han obtenido 90%. Por tanto, estos serán los movimientos que se implementen en el juego final.

En cuanto a los movimientos no aceptados, destacan el 50 y el 33, ambos son detectados fácilmente, sin ningún problema mayor. Sin embargo, sus porcentajes de acierto se encuentran en un 50% y 55% respectivamente. Esto se debe a que hay que realizar el movimiento 5 veces a máxima velocidad, pero no siempre se detectan cinco, lo cual significa que, probablemente, debido a la velocidad, las mediciones no sean lo suficientemente correctas. En cuanto al ítem 45, el problema surge cuando para obtener una puntuación de 1, hay que realizar el movimiento parcialmente, y, por tanto, es necesario detectar una flexión de cadera. Esto es tremendamente inexacto, ya que depende de múltiples factores como la ropa o la estructura de cada persona. Por lo tanto, ya que solo se podría detectar el realizarla correctamente o no, no cumple los mínimos para su implementación. Por último, el ítem 16 es detectado correctamente un 85% de las veces. El porcentaje no es lo suficientemente alto, así que no será implementado. Sin embargo, está situado muy cerca del límite, por lo que, si se mejorase la precisión mediante, por ejemplo, un filtro, podría incorporarse en un futuro.

Finalmente, tras haber analizado la totalidad de ítems de la escala, se ha concluido con una lista de movimientos que se implementarán en el *Serious Game*. En la **tabla 6.9**. se puede observar la lista completa de movimientos adecuados para su realización. Cabe remarcar que la forma de implementarlos ya fue explicada en el capítulo 5.

Extremidad superior (posición sentada)					
N.º Ítem	Movimiento evaluado	Explicación de la puntuación obtenida	Puntuación		
			Fallo	Parcial	Correcto
6	Rotación externa del hombro	Rotación externa del hombro, con el codo a 90º	0	1	2
7	Flexión del codo	El paciente debe tocarse la oreja del lado con el que realice el movimiento	0	1	2
9	Rotación interna del hombro	Rotación interna del hombro, con el codo a 90º	0	1	2
10	Extensión del codo	El paciente es capaz de estirar el codo a 0º con el hombro a 90º.	0	1	2
13	Elevación frontal del hombro	Flexión del hombro, de tal forma que este quede a 90º. El codo ha de mantenerse a 0º.	0	1	2
15	Elevación lateral del hombro	Abducción del hombro a 90º, manteniendo el codo a 0º.	0	1	2

Tabla 6.9: Ítems implementados en el *Serious Game*

7 Entorno socio-económico

En el siguiente capítulo se analizará el impacto social, económico y ético que podría tener el proyecto si se comercializa. Además, se realizará un presupuesto del coste del proyecto y un plan de explotación del mismo.

7.1 Presupuesto

En primer lugar, se pretende realizar un acercamiento al coste del proyecto realizado si se hubiera desarrollado en un ambiente laboral. Para ello, hay que tener en cuenta tanto los costes directos como los indirectos.

Se consideran costes directos a aquellos costes que se relacionen de forma clara con el producto, sin necesidad de reparto. Los más comunes son la mano de obra, la materia prima, los gastos de transporte y los gastos de amortización. En nuestro caso, hay que considerar la mano de obra de una sola persona, con un nivel profesional equivalente al de ingeniero junior, y la amortización del hardware y software utilizado.

Los costes indirectos son los que derivan de la realización del proyecto. Ejemplos de este tipo de costes son las comunicaciones, los almacenes, tiempos perdidos, administración o los imprevistos. Los principales relacionados con nuestro proyecto serían el uso del laboratorio y los imprevistos. Basándose en varias reseñas, estos gastos serán fijados en un 10% respecto al coste total.

Teniendo esto en cuenta se procede a realizar una estimación del coste de la mano de obra. Con este fin, se realizará una suma de las horas dedicadas a cada tarea.

Fase del proyecto	Tarea	Horas empleadas
Documentación	Aprendizaje del lenguaje de programación C# y del uso apropiado de Unity	50
	Implementación de Kinect en Unity	15
	Aprendizaje de funciones específicas de Unity a lo largo del proyecto	25
	Investigación sobre el test *Fugl-Meyer*	20
Desarrollo del proyecto	Desarrollo de juegos de prueba con elementos aplicables al modelo real	30
	Pruebas sobre la precisión del Kinect y elección de los ítems a implementar	40
	Desarrollo del modelo final	100
Memoria	Investigación sobre otros proyectos y estudios relacionados con este	20
	Escritura de la memoria	120
	Revisión y reuniones con el tutor	5
TOTAL		425

Tabla 7.1: Disección detallada de las horas empleadas

Teniendo en cuenta que el salario promedio de un ingeniero junior es de 10€/h, el precio de la mano de obra ascendería a 4250€.

En cuanto al gasto del equipamiento necesario, este es principalmente el hardware y el software utilizado.

Nombre	Coste (€/unidad)	Unidades	Coste de amortización (€)
MSi GT72 2QE Dominator Pro	1999€	1	84.95
Microsoft Windows 10 Pro	279€	1	13.95
Microsoft Sensor Kinect 2.0	100€	1	4
Adaptador del sensor Kinect	42€	1	1.6
Unity y *kinect for windows sdk*	-	1	-
			104.5

Tabla 7.2: Costes del equipamiento

El coste de amortización viene determinado por la cuota de amortización mensual. Esta se calcula dividiendo el valor de adquisición del producto menos su valor residual, y dividiendo el resultado entre el periodo de depreciación, el cual se considerará de 60 meses. Posteriormente, se multiplica ese valor por el número de meses que se ha utilizado y se obtiene el coste de amortización.

Finalmente, el coste total estimado del proyecto se resume en la siguiente tabla.

Concepto	Coste (€)
Recursos Humanos	4250
Amortización	104.5
Costes indirectos	435.45
Coste Total del Proyecto	**4789.95**

Tabla 7.3: Presupuesto final

7.2 Impacto socio-económico

Antes de analizar cuál será, es necesario saber que es el impacto socio-económico.

Se considera un impacto cuando surge un cambio en la vida de la población *target*. Esto puede incluso extenderse a generaciones futuras. Por tanto, los tipos más destacados suelen ser los relacionados con cambios en el nivel educativo, el estado de salud o el nivel de ingresos. En nuestro caso, el proyecto está orientado principalmente a mejorar la salud, sin embargo, también hay un fuerte componente económico.

Teniendo esto en cuenta, se podría decir que el proyecto tiene una propuesta de valor ciertamente robusta. Por un lado, permite una mejora de la salud del usuario. Esta mejora no se remite solamente a personas que han sufrido un ictus, cualquier persona con problemas de movilidad podría usarlo como método de rehabilitación. Si a esto se le añade el indudable atractivo para los jóvenes, por tratarse de un videojuego, y la saciedad en la curiosidad de los adultos por probar una tecnología novedosa, podría llegar a ser un producto muy solicitado. Por otro lado, permite un ahorro en costes a la institución médica que lo utilice. Esto se debe a que, actualmente, este tipo de rehabilitación se realiza siempre con un médico al lado, el cual podría ser relegado de estas tareas, en las que su conocimiento no es estrictamente necesario. En adición, se encuentra el aspecto de los datos que obtiene y almacena. Hoy en día, en la mayoría de las terapias de rehabilitación no se obtienen datos, y si se hace, se adquieren mediante sensores decenas de veces más caros que el propuesto. Por tanto, una propuesta económica como esta podría ganarse un hueco en el mercado.

Sin embargo, todo esto levanta un debate moral. Como es de correcto que la rehabilitación deje de realizarse con un médico. No es necesario también la calidez humana para superar con éxito un periodo de malestar como puede ser el haber sufrido un ictus. Este es un tema muy complejo, en el que por seguro existirían detractores y partidarios. No obstante, este no es motivo como para frenar el proyecto, ya que siempre existe la opción de elegir si usarlo o no.

En cuanto a la explotación del proyecto, este tiene una finalidad claramente orientada hacia la comercialización. Para ello, se vendería el resultado como un videojuego normal para ordenador, con la adición de que este necesita el sensor Kinect para usarse.

Se podría considerar como *target* cualquier institución, empresa o particular, dedicada al tratamiento de pacientes con alguna discapacidad física que requiera de rehabilitación. Por lo tanto, en caso de ser comercializado, el producto sería vendido a clínicas privadas y sistemas gubernamentales principalmente, aunque también puede haber un público más reducido que lo adquiera, como médicos particulares o los propios pacientes.

Finalmente, para concluir remarcar el hecho de que es un producto con capacidad de comercialización, centrado en mejorar la salud de las personas mientras mantiene un costo relativamente pequeño en comparación a otras tecnologías existentes.

8 Conclusiones y líneas futuras

Este último apartado analizará el trabajo realizado y los resultados finales obtenidos. Esto permitirá conocer que grado de realización de los objetivos iniciales se ha conseguido. Además, se tratarán las posibles mejoras del proyecto y su posible futuro, ya que se trata de un territorio ampliamente inexplorado en el que se pueden lograr una gran cantidad de avances. Por último, se mencionarán los mayores problemas encontrados durante la realización del proyecto.

8.1 Conclusiones

El principal objetivo del proyecto era crear una aplicación que ayudase a personas con problemas motores en su rehabilitación. Concretamente, se centró en las que han sufrido un ictus, aunque el resultado del proyecto es extrapolable a otras patologías, es decir, cualquier persona puede usarlo como parte de su rehabilitación, e incluso como diversión. Si nos centramos en los objetivos logrados, se puede destacar la implementación de varios de los ítems que componen el test *Fugl-Meyer*, la creación de un minijuego para que el paciente pueda entrenar su movilidad y el reducido coste en comparación con otros sistemas existentes en el mercado.

Por tanto, se podría considerar que a grandes rasgos se han cumplido los objetivos del proyecto. Sin embargo, estos son todavía mejorables. En comparación a la gran cantidad de ítems existentes en el *Fugl-Meyer*, la cantidad desarrollada es muy pequeña. Esto se ha debido a falta de precisión en el sensor, movimientos que ponen en riesgo la integridad del paciente y elementos que necesitan para su realización la intervención de una persona. En este sentido, la evaluación realizada no se encuentra al mismo nivel que la que podría realizar un terapeuta. Por lo tanto, el resultado del proyecto no se puede considerar como una herramienta que libere al especialista de ciertas de sus actividades, sino que finalmente, se trata de un elemento de apoyo con el que el paciente pueda salir de la rutina, motivarse y seguir un progreso más preciso de sus resultados.

En cuanto a estos, la adquisición y guardado de datos se ha realizado tal y como estaba planteada desde un principio, por lo que, en este caso, si que se podría considerar un éxito. El programa genera una serie de archivos CSV en los que se encuentra toda la información registrada. Por un lado, se guardan los datos obtenidos con las mediciones del *Kinect*, es decir, la posición, la orientación y el ángulo de las articulaciones. Toda esta información se planea que será analizada en un futuro mediante *Matlab* para poder realizar un análisis más preciso de los movimientos registrados, obteniéndose patrones o tendencias, por ejemplo. Por otro lado, se almacenan todos los datos de la sesión, esto es la progresión directa del paciente en el videojuego. Esta se detalla mediante la puntuación, los fallos y los tiempos obtenidos en cada uno de los apartados. Por tanto, se puede considerar que el apartado de la información ha cumplido sobradamente con su objetivo.

En el apartado de la variedad jugable, se ha encontrado una gran limitación. Inicialmente, se pensaban crear una gran cantidad de minijuegos en los que el paciente pudiese entrenar las distintas partes del cuerpo de diferentes formas. No obstante, esta línea de trabajo fue suprimida en favor de otras como el análisis de la precisión y los mejores planos de medida del *Kinect* o el dar la posibilidad al terapeuta de poder personalizar las sesiones lo máximo posible. Como consecuencia, el resultado final no posee una gran variedad de variaciones de entrenamiento, dando en su lugar una personalización de las existentes y un análisis de las mediciones del sensor utilizado.

Si nos referimos a este, tras finalizar el proyecto se puede decir que la elección de esta tecnología fue la correcta, ya que cuenta con una documentación y un apoyo detrás mucho más grande que el de otros sensores, lo cual ha servido para solucionar una gran cantidad de problemas. Además, esto ha permitido que el proyecto cumpla otro de sus grandes objetivos, un bajo coste. Finalmente, se han realizado unas medidas adecuadas con un coste menor al de otros equipos, lo que sin lugar a dudas, favorece el crecimiento de este tipo de proyectos.

8.2 Líneas futuras

Basándose en las conclusiones que se acaban de exponer, las líneas de trabajo a seguir están bastante claras. Recordemos que el objetivo es que sea una herramienta capaz de sustituir al especialista en una rehabilitación. Para lograr este objetivo, hay que trabajar, en primer lugar, en un menú mucho más amplio y específico que el existente, que ofrezca todas las opciones que podría ofrecer una persona. Para aclarar este concepto se formularán algunos ejemplos: Durante una tarea el paciente puede cansarse, sufrir un mareo o simplemente despistarse, conllevando esto un fallo en la prueba que esté realizando. Por tanto, sería necesario un menú de pausa que permitiese volver a un punto específico del juego. Otra situación probable sería la de querer cambiar los parámetros de la sesión mientras esta se realiza, algo que actualmente solo se puede hacer antes o después. También se podría dar el caso, de que se quisiera un programa autónomo, que el paciente pudiese realizar por su cuenta. Para ello, se podría hacer que la mano del paciente moviese el ratón del ordenador y que cerrar el puño equivaliese a un clic. En esta línea de trabajo se podría determinar una posición inicial, mediante la cual el propio paciente pudiese iniciar los distintos minijuegos. Así, se podría continuar hasta que se ultimasen todos los detalles de la mayor cantidad de situaciones probables.

Otro aspecto importante que se puede mejorar es la evaluación del *Fugl-Meyer*. Con tiempo suficiente se podrían desarrollar algoritmos o filtros capaces de solucionar el problema de la precisión en gran cantidad de movimientos. Además, se podría introducir el uso de robots para realizar los ítems en los que se requiere una persona. Esto es, por ejemplo, cuando se realiza una prueba de agarre, en la que se evalúa la fuerza del paciente. En el proyecto desarrollado esto no se puede implementar, pero si se fusiona con otros proyectos realizados en el laboratorio el resultado podría ser la inclusión de más ítems en el *Serious Game*. De hecho, es muy probable que, si se le dedica los medios necesarios, se podría realizar el test por completo de forma autónoma.

Finalmente, se podría incluir una gran cantidad de minijuegos, que se dedicasen a entrenar las debilidades de los distintos pacientes. Se podrían personalizar de tal forma, que cada paciente pudiese tener un esquema, en el que se mostrase que ejercicios realizar cada día, con que parámetros y, en general, todo lo que necesitase saber. Pero no solo eso, algunos de estos nuevos ejercicios pueden ser otros test u otro tipo de evaluaciones. De forma que el programa se convirtiese en un *software* automatizado con una gran cantidad de evaluaciones y ejercicios, destinados a abarcar la mayor cantidad de pacientes posibles.

8.3 Problemas enfrentados

Hay que destacar que se han contado con varios problemas graves durante el transcurso del proyecto. En primer lugar, hay que destacar la comunicación entre el *Kinect* y *Unity*. En la etapa más temprana del proyecto se buscó una gran cantidad de información, pero nada prometía un buen resultado. La mayoría de los tutoriales e hilos en foros que tratan el tema son muy escuetos, sin explicaciones claras y centrados totalmente en los drivers oficiales de *Microsoft* contenidos en el paquete *Kinect for Windows*. No fue hasta que se investigó en la *Asset Store*, cuando se encontraron las herramientas necesarias para realizar una implementación correcta. Por tanto, se puede decir, que la información al respecto es demasiado escasa.

En segundo lugar, hay que destacar la propia conexión del *Kinect* con el ordenador. Esta no es complicada en sí misma, ya que con tener los drivers instalados y el Kinect conectado mediante el USB todo debería de funcionar. Sin embargo, existen una gran cantidad de errores reportados, principalmente por la compatibilidad con tipos específicos de placas base, tarjetas gráficas y puertos USB. En mi caso en particular, intenté conectar el Kinect con dos portátiles distintos, uno de ellos totalmente nuevo, y en ninguno de ellos llegó a funcionar. En ambos casos hubo problemas con la detección del USB 3.0, a pesar de que ambos lo tenían. Por suerte, el ordenador del laboratorio de robótica asistencial si que se reconoce correctamente.

Por último, destacar la implementación de ítems del *Fugl-Meyer* en el *Serious Game*. La elección del método para valorar cada movimiento fue muy compleja, ya que Unity está enfocado al desarrollo de videojuegos, no a este tipo de aplicaciones en las que se obtienen y analizan datos. Por ello, hubo que adaptarse a las herramientas ofrecidas para que estas pudiesen ser usadas para la evaluación del paciente. En este tema, también destacar la dificultad de implementar movimientos como los que conforman el *Fugl-Meyer* de forma automática, ya que están diseñados para que un especialista los explique y ayude al paciente a comprenderlos.

Como conclusión final se podría añadir que este proyecto es solo el comienzo de lo que podría ser como una revolución en la rehabilitación. Es evidente que la tecnología cada vez está más presente en nuestras vidas y que su principal uso es el de mejorar la calidad de vida. Por tanto, que sea usada en este ámbito es solo un paso evidente que se dará tarde o temprano.

Bibliografía

[1] Elotrolado.net (2017). *Historia de los Videojuegos: Los inicios*, ElOtroLado. [Online]. Disponible en: https://www.elotrolado.net/wiki/Historia_de_los_videojuegos:_Los_inicios [Accedido el 21 de agosto de 2017]

[2] Historia de los Videojuegos (s.f.). En Wikipedia. Recuperado el 21 de agosto de 2017 de https://es.wikipedia.org/wiki/Historia_de_los_videojuegos

[3] Belli, S. y López Raventós, C. (otoño 2008). *Breve historia de los videojuegos*. Athenea Digital (14), p159-179.

[4] Fernández San Román, M. (junio 2015). *Juego Serio para estimulación física y cognitiva* (Trabajo Fin de Grado). Universidad de Valladolid, Valladolid, España.

[5] González Muñoz C., Gracia Bandrés, M.A., Sanagustín Grasa, L., Romero San Martín, D. – (2015) *TecsMedia: Análisis Motores gráficos y su aplicación en la industria.*

[6] Sitzmann, T. (octubre 2010*). Game on?: The effectiveness of game use in the workplace depends on context and design*. Talent Development. Volumen 64 (10), p20.

[7] Morales Moras, J. San Cornelio Esquerdo, G. (noviembre 2016). *La jugabilidad educativa en los Serious Games: pautas para el diseño de videojuegos con un propósito educativo y de cambio social*. Artediez (10) p1-23.

[8] Ferrer González, B.M. (2016). *Adaptación y validación al español de la escala Fugl-Meyer en el manejo de la rehabilitación de pacientes con ictus*. (Tesis doctoral inédita). Universidad de Sevilla, Sevilla, España.

[9] García Escrivá, S. (agosto 2017). *Ictus*. Webconsultas. Recuperado el 26 de agosto de 2017 de http://www.webconsultas.com/ictus/ictus-582.

[10] Arias, A. (2014). *Aprende a Desarrollar Videojuegos*. CreateSpace Independent Publishing Plarform.

[11] Berzal F. (s.f.) *Desarrollo de videojuegos.* [Online] Departamento de Ciencias de la computación e I.A. Universidad de Granada, Granada. Recuperado el 6 de agosto de 2017 de http://elvex.ugr.es/decsai/games/slides/lab/P1-Herramientas.pdf

[12] Adobe System Software. (2017). *Adobe AIR*. Adobe. Recuperado el 8 de agosto de 2017 de http://www.adobe.com/devnet/games/gaming.html

[13] Navarro, A. Pradilla, J.V. Ríos, O. (2012). *Open Source 3d Game Engines for Serious Games Modeling.* Prof. Caralin Alexandru (Ed.), InTech. Disponible en: http://www.intechopen.com/books/modeling-and-simulationinengineering/open-source-3d-game-engines-for-serious-games-modeling Accedido el 7 de agosto de 2017.

[14] YoYo Games Ltd., (2015). *Working with 3D.* Yoyogames. Recuperado el 7 de agosto de 2017 de https://docs.yoyogames.com/source/dadiospice/002_reference/drawing/drawing%203d/index.html

[15] GameMaker: Studio. (2017). En Wikipedia. Recuperado el 7 de agosto de 2017 de https://es.wikipedia.org/wiki/GameMaker:_Studio

[16] Unity Technologies. (2017) *Unity 2017: El motor de creación de juegos líder en el mundo.* Recuperado el 5 de agosto de 2017 de https://unity3d.com/es/unity

[17] Lozano Ortega, M.A. (2016). *Motores para videojuegos: ¿Cuál elegir?.* Departamento de Ciencia de la Computación e I.A. Universidad de Alicante. Recuperado el 6 de agosto de 2017 de https://moodle2015-16.ua.es/moodle/pluginfile.php/91101/mod_resource/content/1/jornada-sgae.pdf

[18] Develop, (2014). *The Tech List*. develop-online.net. Recuperado el 5 de agosto de 2017 de http://content.yudu.com/A2xcc7/Dev100TechList2014/resources/index.htm?referrerUrl

[19] Unity Techonologies (2016). *El software líder a nivel mundial en la industria de los juegos.* Recuperado el 6 de julio de 2017 de https://unity3d.com/es/public-relations

[20] Gregory, J. (2009) *Game Engine Architecture*. Miami, EE. UU. A K Peters, Ltd.

[21] Epic Game ,Inc. (2017) *Frequently asked questions (FAQ).* Recuperado el 7 de agosto de 2017 https://www.unrealengine.com/faq

[22] CryEngine. (2017). En Wikipedia. Recuperado el 7 de agosto de 2017 de https://en.wikipedia.org/wiki/CryEngine

[23] Crytek GmbH. (2017) *Get Cryengine*. Recuperado el 7 de agosto de 2017 de https://www.cryengine.com/get-cryengine

[24] Crytek GmbH. (2017) *Features*. Recuperado el 7 de agosto de 2017 de https://www.cryengine.com/features/visuals

[25] Pérez Romero, F.M. (julio 2011). *Estudio de técnicas de animación para personajes de videojuegos.* (Trabajo Fin de Master). Escuela Técnica superior de ingeniería informática, Sevilla, España.

[26] Motion Capture. (2013). En Wikipedia. Recuperado el 9 de agosto de 2017 de https://en.wikipedia.org/wiki/Motion_capture

[27] Fernández Carrasco, F. Botana Gómez, J. Sempere Tortosa, M. González Gómez, C. Navarro Soria, I.J. Gil Méndez, D. (2014). *Captura de movimiento del alumnado: una aproximación*. Universidad de Alicante, Alicante, España.

[28] Microsoft. (2015). *Kinect for Windows Sensor Components and Specifications*. Recuperado el 11 de agosto de 2017 de: https://msdn.microsoft.com/en-us/library/jj131033.aspx

[29] Microsoft. (2015). *Kinect for Windows SDK*. Recuperado el 11 de agosto de 2017 de: https://msdn.microsoft.com/en-us/library/dn799271.aspx

[30] Softkinetic. (s.f.) *Products*. Recuperado el 13 de agosto de 2017 de https://www.softkinetic.com/Products

[31] ASUSTeK Computer Inc. (s.f.) *Xtion PRO LIVE*. Recuperado el 13 de agosto de 2017 de https://www.asus.com/es/3D-Sensor/Xtion_PRO_LIVE/

[32] Creative Technology Ltd. (s.f.). *Creative Senz3D*. Recuperado el 13 de agosto de 2017 de https://us.creative.com/p/web-cameras/creative-senz3d

[33] Livingstone, S.R. (octubre 2008). *Macquarie Motion Capture Manual. Understanding Vicon Nexus*. Recuperado el 11 de agosto de 2017 de http://psy.mq.edu.au/me2/mocap_v1/introduction.html

[34] Statista. (2014). *What game engines do you currently use?*. Recuperado el 15 de agosto de 2017 de https://www.statista.com/statistics/321059/game-engines-used-by-video-game-developers-uk/

[35] Herreros Díaz, J. (octubre 2016). *Implementation of Open Source applications "Serious Game" for rehabilitation*. (Trabajo Fin de Grado). Universidad Carlos III de Madrid, Madrid, España.

[36] Steward, J. Llichti, D. Chow, J. Ferber, R. & Osis, S. (mayo 2015). *Performance assessment and Calibration of the Kinect 2.0 Time-of-Flight: Range Camera for Use in Motion Capture aplications*. University of Calgary, Canada.

[37] Wasenmüller, G. Stricker, D. (2016) *Comparison of Kinect v1 and v2 Depth Images in Terms of Accuracy and Precision*. German Research Center for Artificial Intelligence (DFKI), Alemania.

ANEXO

GetJointPosition.cs

Este *script* se encarga de guardar la posición en tres dimensiones de la articulación que se le pida. Simplemente hay que introducir la articulación a analizar, si detecta a un usuario le rastrea la susodicha articulación y obtiene su posición. Además, también guarda esto en formato CSV. En primer lugar, crea el archivo en la dirección que se le indique y pone en la primera línea time ,pos_x, pos_y, pos_z. A continuación, por cada *frame*, va completando las siguientes líneas con los valores correspondientes.

```
using UnityEngine;
using System.Collections;
using System.IO;

public class GetJointPosition : MonoBehaviour {
    [Tooltip("The Kinect joint we want to track.")]
    public KinectInterop.JointType joint = KinectInterop.JointType.HandRight;

    [Tooltip("Current joint position in Kinect coordinates (meters).")]
    public Vector3 jointPosition;

    [Tooltip("Path to the CSV file, we want to save the joint data to.")]
    private string saveFilePath;

    public string file_name;
    private float tiempo = 0.0f;

    void Start()
    {

        saveFilePath = file_name + "Position_" +
((KinectInterop.JointType)joint).ToString() + ".csv";

        if (!File.Exists(saveFilePath))
        {
            using (StreamWriter writer = File.CreateText(saveFilePath))
            {
                string sLine = "time,pos_x,pos_y,poz_z";
                writer.WriteLine(sLine);
            }
        }
    }

    void Update()
    {
        KinectManager manager = KinectManager.Instance;

        if (manager && manager.IsInitialized())
        {
            if (manager.IsUserDetected())
            {
                long userId = manager.GetPrimaryUserID();

                if (manager.IsJointTracked(userId, (int)joint))
                {
                    Vector3 jointPos = manager.GetJointPosition(userId,
(int)joint);
```

```
                    jointPosition = jointPos;
                    tiempo += Time.deltaTime;

                            using (StreamWriter writer =
File.AppendText(saveFilePath))
                            {
                                string sLine =
string.Format("{0:F3},{1},{2:F3},{3:F3},{4:F3}", tiempo, jointPos.x, jointPos.y,
jointPos.z);
                                writer.WriteLine(sLine);
                            }
                }
            }
        }

    }

}
```

Angleplane.cs

Este *scrip*t calcula los ángulos existentes entre dos articulaciones y los ejes de coordenadas, tal y como se explicó anteriormente. Para ello, es necesario introducir las dos articulaciones, y tres "Text" que servirán para mostrar los ángulos obtenidos por pantalla. Una vez que se detecta a un usuario, se obtienen las coordenadas de las articulaciones seleccionadas, se realizan los cálculos para obtener los ángulos y se muestran por pantalla.

```
using UnityEngine;
using System.Collections;
using System.IO;
using UnityEngine.UI;

public class Angleplane : MonoBehaviour
{
    [Tooltip("The Kinect joint we want to track.")]
    public KinectInterop.JointType joint1 = KinectInterop.JointType.SpineMid;
    public KinectInterop.JointType joint2 =KinectInterop.JointType.SpineShoulder;

    [Tooltip("Current joint position in Kinect coordinates (meters).")]
    public Vector3 jointPosition1;
    public Vector3 jointPosition2;
    public Text debugText;
    public Text debugText1;
    public Text debugText2;
    public float anglexy;
    public float anglexz;
    public float angleyz;

    void Update()
    {

        KinectManager manager = KinectManager.Instance;

        if (manager && manager.IsInitialized())
        {
            if (manager.IsUserDetected())
            {
                long userId = manager.GetPrimaryUserID();
```

```
                if (manager.IsJointTracked(userId, (int)joint1) &&
manager.IsJointTracked(userId, (int)joint2))
                {
                    Vector3 jointPos1 = manager.GetJointPosition(userId,
(int)joint1);
                    jointPosition1 = jointPos1;

                    Vector3 jointPos2 = manager.GetJointPosition(userId,
(int)joint2);
                    jointPosition2 = jointPos2;

                    anglexy = Mathf.Atan2(jointPosition2.y - jointPosition1.y,
jointPosition2.x - jointPosition1.x)*Mathf.Rad2Deg;
                    debugText.text = string.Format("{0}", anglexy);

                    angleyz = Mathf.Atan2(jointPosition2.z - jointPosition1.z,
jointPosition2.y - jointPosition1.y) * Mathf.Rad2Deg;
                    debugText1.text = string.Format("{0}", angleyz);

                    anglexz = Mathf.Atan2(jointPosition2.z - jointPosition1.z,
jointPosition2.x - jointPosition1.x) * Mathf.Rad2Deg;
                    debugText2.text = string.Format("{0}", anglexz);
                }
            }
        }
    }
}
```